НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»

Теплоенергетичний факультет

Кафедра автоматизації проектування енергетичних процесів і систем

До захисту допущено
Завідувач кафедри
_____ О.В. Коваль
(підпис) (ініціали, прізвище)
“___”_____2020р.

ДИПЛОМНА РОБОТА

на здобуття ступеня бакалавра

з напряму підготовки 121 Інженерія програмного забезпечення
на тему Аналіз відеопотоку: ідентифікація людей за статтю і віковою групою

Виконав (-ла): студент (-ка) 4 курсу, групи ТВ-61

Герасимова Марія Владиславівна _____ _____
(прізвище, ім’я, по батькові) (підпис)

Керівник професор кафедри АПЕПС, д.е.н Сігайов А. О. _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Рецензент ст. викладач кафедри АЕС і ІТФ, к.т.н Воробйов М.В. _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі
немає запозичень з праць інших авторів
без відповідних посилань.
Студент _____
(підпис)

Київ – 2020 року

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет теплоенергетичний

Кафедра автоматизації проектування енергетичних процесів і систем

Рівень вищої освіти перший рівень

Напрям підготовки: 121 Інженерія програмного забезпечення

Спеціалізація: Програмне забезпечення розподілених систем

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ О.В. Коваль
(підпис)
”___”_____2020р.

ЗАВДАННЯ
на дипломну роботу студенту

Герасимовій Марії Владиславівні
(прізвище, ім’я, по батькові)

1. Тема роботи Аналіз відеопотоку: ідентифікація людей за статтю і віковою групою

керівник роботи Сігайов Андрій Олександрович, д.е.н.
(прізвище, ім’я, по батькові науковий ступінь, вчене звання)

затверджена наказом вищого навчального закладу від ”25” травня 2020р. № 1168-с

2. Строк подання студентом роботи _____

3. Вихідні дані до роботи Мова програмування Python, СУБД MySQL, бібліотеки Keras, OpenCV, Kivy, unittest.

4.Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) Проаналізувати сучасні методи вирішення задачі розпізнавання, обрати архітектуру моделі машинного навчання, знайти і підготувати набір даних, створити і натренувати модель, розробити структуру бази даних системи і реалізувати її, розробити інтерфейс користувача системи, провести збірку і тестування програмного продукту.

5. Перелік ілюстративного матеріалу
“Проблематика теми та актуальність”, “Постановка задачі і аналіз рішень”, “Опис структури системи”, “Представлення результатів аналізу”, “Концептуальна модель бази даних системи”, “Модель машинного

навчання”, “Вибір формату даних”, “Тренування моделі машинного навчання”, “Використані технології”, “Результати роботи системи”, “Результати роботи системи — аналіз розподілу”, “Висновки”.

6. Дата видачі завдання ”11” жовтня 2019 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітки
1.	Затвердження теми роботи	11.10.2019	
2.	Вивчення та аналіз задачі	19.04.2020	
3.	Розробка архітектури та загальної структури системи	26.04.2020	
4.	Розробка структур окремих підсистем	03.05.2020	
5.	Програмна реалізація системи	13.05.2020	
6.	Оформлення пояснювальної записки	08.06.2020	
7.	Захист програмного продукту	09.06.2020	
8.	Передзахист	09.06.2020	
9.	Захист	16.06.2020	

Студент _____ Герасимова М. В.
(підпис) (прізвище та ініціали,)

Керівник роботи _____ Сігайов А. О.
(підпис) (прізвище та ініціали,)

АНОТАЦІЯ

Записка до дипломної роботи на тему “Аналіз відеопотоку: ідентифікація людей за статтю та віковою групою” містить 66 сторінок, 41 рисунок, 3 додатка та 13 бібліографічних найменувань.

Метою дипломної роботи є розробка програмного забезпечення, яке за допомогою моделі машинного навчання приймає рішення про належність до вікової та статевої групи людини, чиє обличчя попадає до зони видимості камери. За прийняття рішень у системі відповідає згорткова нейронна мережа — модель машинного навчання, що надає найкращі показники метрик якості у роботі з відеоматеріалами та зображеннями. В процесі виконання роботи проведена обробка даних, спроектовано нейронну мережу зі складною архітектурою, проведено її навчання, реалізовано інтерфейс системи для обробки даних, що отримуються в результаті роботи системи.

ABSTRACT

The note of diploma thesis on "Video Stream Analysis: Identification of People by Sex and Age Group" contains 64 pages, 20 figures, 3 appendices and 13 bibliographic titles.

The purpose of the thesis is to develop software that uses a model of machine learning to decide whether to belong to the age and gender group of a person whose face falls within the field of view of the camera. The convolutional neural network is responsible for decision-making in the system — a model of machine learning that provides the best indicators of quality metrics in working with video and images. In the course of performance of work data processing is carried out, the neural network with difficult architecture is designed, its training is carried out, the system interface for processing of the data received as a result of work of system is realized.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

1. CNN — Convolutional Neural Network, згорткова нейронна мережа — клас нейронних мереж, в основу архітектури якої покладена операція згортки.

2. MNIST — Modified National Institute of Standards and Technology — назва для набору даних рукописних чисел.

3. CPU — Central processing unit, центральний процесор, який представляє собою функціональну частину комп’ютера.

4. GPU — Graphics Proccesing Unit, графічний процесор комп’ютера.

5. UI — User Interface, інтерфейс користувача.

6. ПЗ — програмне забезпечення.

7. kvlang — Kivy language, спеціальна мова, що створена для написання інтерфейсу користувача за допомогою бібліотеки Kivy.

8. ReLU — rectified linear unit — функція активації для нейронних мереж — кусково-лінійна функція, яка буде виводити вхід безпосередньо, якщо він позитивний, в іншому випадку, вона буде виводити нуль.

9. ResNet — Residual Network, залишкова мережа — клас нейронних мереж, в основі архітектури яких лежать залишкові блоки зі зв’язками оригінального входа блоку з виходом шарів блока.

ЗМІСТ

ВСТУП

Системи ропізнавання та ідентифікації широко розповсюджуються у більшості галузей: охоронній, фінансовій, промисловій, розважальній тощо, виконують важливі функції і мають велике майбутнє у розвитку. Усі переваги вони отримують як категорія систем штучного інтелекту, які не мають своїх аналогів по ефективності в інших сферах інформаційних технологій без використання машинного навчання.

Розпізнавання відео є невід'ємною частиною аналізу відеоматеріалів. Воно допомагає знайти сенс в безлічі візуальних даних, з якими люди мають справу щодня. Алгоритми, що виконують ці задачі, об’єднані зі спорідненими під назвою “комп’ютерний зір”.

Комп'ютерний зір - область штучного інтелекту, що навчає комп'ютерні програми інтерпретувати і розуміти візуальний світ. Використовуючи цифрові зображення з камер, відео та моделі глибокого навчання, комп'ютери точно ідентифікують і класифікують об'єкти, а потім реагують, коли бачать їх знову. Прикладом комп'ютерного зору є ідентифікація облич на фото чи з камери відеоспостереження, ідентифікація транспортних засобів на дорозі, або ж класифікація людей за певними ознаками, найпростіші з них — вік та стать. Системи ідентифікації стануть у нагоді будь-яким додаткам, що використовують відеопотоки, та охоронним компаніям для пошуку людей або регулювання стану на вулицях і в приміщеннях.

Сучасні установи для вирішення проблем безпеки потребують кілька спеціально підготовлених співробітників. Ці співробітники роблять помилки, які можуть вплинути на рівень безпеки. Таку проблему вирішує створення системи розпізнавання і обробки відеоматеріалів.

Автоматична класифікація за віком і статтю стає все більш актуальною для зростаючого числа додатків, особливо в зв'язку з ростом числа

соціальних платформ і соціальних мереж. Не всі додатки вимагають від користувача внесення особистих даних, лише окремі категорії додатків дійсно потребують ці дані для використання користувачами основного або важливого функціоналу системи, і незначна кількість з них дійсно отримує ці дані. Якщо ж розглянути питання коректності введених даних користувачем про себе, результати не задовольнять аналітиків. Але значна більшість додатків потребують знання про розподіл множини своїх користувачів. Розроблений в межах дипломної роботи програмний продукт представляє собою мультифункціональну систему, що повністю вирішує поставлену проблему, не обмежуючись однією задачею розпізнавання відеопотоку.

1. ЗАДАЧА ІДЕНТИФІКАЦІЇ ЛЮДЕЙ ПІД ЧАС АНАЛІЗУ ВІДЕОПОТОКУ

Цільова система за темою дипломної роботи представляє собою програмне забезпечення для аналізу візуальних даних у вигляді відеопотоку. Аналіз має за мету ідентифікацію людей, зображених на відео, класифікуючи за статевою та віковою ознаками. Подібну задачу ідентифікації виконують моделі машинного навчання. Створення такої моделі передбачає виконання декількох задач, серед них — збір і підготовка даних та вибір архітектури моделі.

Така система матиме можливість збирати відомості про вік і стать користувача камерою, тож необхідна функція авторизації і зберігання персональних даних у базу. Дані про користувачів можна не тільки зберігати, але й аналізувати, отримуючи інформацію про розподіл за віком і статтю.

1.1. Аналіз відеоматеріалів з метою розпізнавання

Технологія розпізнавання відео глибоко занурюється в відеоконтент для його ідентифікації. Вона ретельно аналізує відеопотік (в режимі реального часу або в автономному режимі) з кадрів для створення корисної інформації про контент.

Ідентицікація людей чи об’єктів з відеопотоку насправді зводиться до багаторазової обробки і аналізу зображень — що є кадрами, з яких складається певне відео. Розпізнавання зображень — це здатність системи або програмного забезпечення ідентифікувати об'єкти, людей, місця і дії на зображеннях.

Використання алгоритмів машинного навчання з вчителем передбачає пошук даних, подібних тим, що будуть обролятися системою для отримання

відповіді про розпізнавання ознак віку і статі людини по зображенню обличчя. Такий набір даних називають датасетом. Якість роботи системи залежатиме від якості датасету, тож наступною необхідною задачею є обробка даних (зображень) перед поданням їх на навчання моделі.

Насправді, розпізнавання відео нагадує людський зір, коли ми відразу розпізнаємо об'єкти в потоці пов'язаних зображень, які бачимо, і визначаємо, що відбувається навколо нас. Тому не дивно, що технологія розпізнавання відео тепер широко використовується в різних областях. До них відносяться охорона і безпека, охорона здоров'я, транспорт, роздрібна торгівля, розваги і домашня автоматизація.

1.2. Ідентифікація у системах машинного навчання

Коли мова заходить про ідентифікацію зображень люди можуть чітко розпізнавати і розрізняти різні особливості об'єктів. Це відбувається тому, що наш мозок був навчений несвідомому використання одного і того ж набору образів, що призвело до розвитку здібності легко розрізняти речі. Ми навряд чи усвідомлюємо, коли інтерпретуємо реальний світ. На відміну від людського мозку, комп'ютер сприймає зір як масив числових значень і шукає закономірності в цифровому зображенні, неважливо чи воно нерухоме, графічне, живе чи відео, щоб розпізнати і розрізнити ключові особливості зображення.

Комп'ютер використовує технології машинного зору з штучним інтелектом і навчені алгоритми розпізнавання зображень через систему камер для аналізу і розуміння візуальних образів з одного зображення або послідовності зображень. Спосіб, яким система інтерпретує зображення, повністю відрізняється від людського. Моделі, що працюють в основі системи комп'ютерного зору розрізняють багато різних частин зображення, ідентифікують межі і потім моделюють підкомпоненти. Використовуючи

фільтрацію і ряд дій через глибокі мережеві шари, вони можуть об'єднати всі частини зображення для генерації висновку.

Такий складний математичний процес має і складну структуру. Відрізняють безліч архітектур для моделей машинного навчання, що частіше використовуються для обробки зображень — нейронних мережей. Тож підбір архітектури мережі відіграє ключову роль у створенні системи машиного навчання. Підібравши архітектуру, модель необхідно безпосередньо створити, далі перед і під час навчання моделі налагоджувати її роботу параметрами.

1.3. Аналіз розподілу даних

Коли дані з відеопотоку отримані, вони потребують обробки. Таким чином система зможе запропонувати повний функціонал для роботи з даними, що складатиметься з їх отримання, конвертування, обробки і аналізу, представлення результату в необхідному і зрозумілому користувачу (тут користувач — власник додатка, що хоче оцінити потреби своїх клієнтів) форматі.

Відомості про вік та стать людей, зображених на відеоматеріалах (відеопотоках, що обробляються в режимі реального часу і не зберігаються) необхідно внести у базу даних. Таким чином модель бази даних має повністю відповідати усім вимогам і забезпечувати налагоджену роботу програмного забезпечення і взаємодію з іншими його компонентами. Тоді база даних зберігає деяку вибірку людей. Мінімум за двома ознаками (вік та стать) можна вже побудувати розподіл, відшукати певні статистики, в яких власники додатку будуть зацікавлені, а саме розподіл кожної змінної — розподіл по віку, розподіл по статі, оцінити взаємозв’язок змінних. Додавши нові змінні, певні дати, що забезпечують інформацією про використання користувачами додатка, статистичний аналіз розширюється.

1.4. Розробка користувацького інтерфейсу користувача

Система, що здатна розпізнавати вік і стать на зображенні, повинна мати інтерфейс для інтерактивного спілкування з користувачем, що дозволятиме отримувати зображення з камери. Для ідентифікації користувача необхідна авторизація — для кожного буде відповідний запис у базі даних, що гарантуватиме налагоджену роботу системи. Відповідно до даних авторизації у базі має оновлюватись інформація про час входу користувача в систему, що також може бути використано для аналізу поведінки користувача. Індивідуальні дані, такі як пароль, мають бути захищені як під час зберігання, так і при передачі їх між системою і базою. Доступ до бази даних повинен мати адміністратор через додаток. Таким чином він зможе редагувати вміст бази даних і переглядати результати аналізу розподілу даних, представлені у вигляді графіків. Функціонал додатка і вигляд інтерфейсу користувача мають бути візуально інтуїтивно зрозумілі і зручні у використанні.

2. СУЧАСНІ ВИРІШЕННЯ ЗАДАЧІ РОЗПІЗНАВАННЯ

Для реалізації задачі, поставленої в Розділі 1, необхідний алгоритм машинного навчання, а саме глибокого навчання. Нейронні мережі і глибоке навчання в даний час забезпечують найкращі рішення багатьох проблем в області розпізнавання образів, розпізнавання мови і обробки природної мови.

2.1. Поняття штучного інтелекту і машинного навчання

Скорочено область штучного інтелекту можна визначити як автоматизацію інтелектуальних задач, що зазвичай виконуються людьми [1]. Тож штучний інтелект — це область, що охоплює машинне навчання і глибоке навчання, а також включає багато підходів, що не пов’язані з навчанням (Рисунок 2.1).

Наприклад, перші програми для гри в шахмати використовували тільки чітко визначені правила, втановлені програмістами, тож не могли кваліфікуватись як машинне навчання. Такий підхід відомий як символічний штучний інтелект, і протягом довгого періоду часу експерти вірили, що досягнути створення штучного інтелекту рівня людини можна використовуючи достатньо складну систему правил.

В другій половині XX століття розвивалась одна з областей штучного інтелекту, що описує розробку алгоритмів, які навчаються самостійно, отримуючи знання з даних для створення прогнозів. І ця область отримала назву машинне навчання. Системі, що працює за цією методологією, передається набір прикладів вирішення певної задачі або лише прикладів, а вона знаходить статистичну структуру, що визначатиме правила для автоматичного знаходження відповіді на поставлене питання (наприклад,

система може віднайти статистичні правила для класифікації фото на певну тематику, якщо представити їй набір таких фото).

Рисунок 2.1 — Поняття штучного інтелекту, машинного і глибокого навчання

На Рисунку 2.2 представлена схематична різниця міє класичним програмуванням — символічним штучним інтелектом та машинним навчанням.

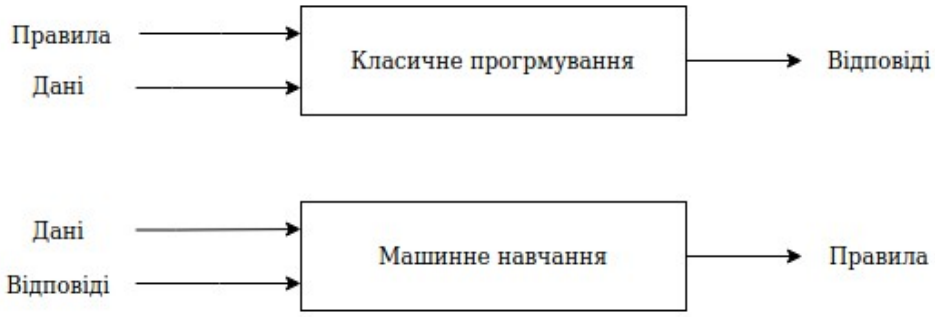


Рисунок 2.2 — Різниця між класичним програмуванням і машинним навчанням

Машинне навчання тісно пов’язане з математичною статистикою, але

зазвичай працює з великим наборами даних, для яких неможливо застосувати класичні методи програмування. Тож робота таких систем в основному базується на інженерних рішеннях, і не доводяться теоретично, не маючи сильної математичної платформи.

2.2. Опис роботи алгоритмів машинного навчання

Щоб зрозуміти поняття глибокого навчання, щоб відрізнити його від поняття машинного навчання, потрібно отримати представлення, чим же займаються алгоритми машинного навчання і як.

В першу чергу, варто описати розподіл машинного навчання на три типи: навчання з вчителем, навчання без вчителя, навчання з підкріпленням. Під „вчителем“ мають на увазі представлені моделі приклади результатів, за якими вона може навчитися формувати власні вірні відповіді. Як було вказано, машинному навчанню потрібні дані — набір прикладів. При навчанні з вчителем алгоритм має доступ до відповідей по цим прикладам, при навчанні без вчителя — система навчається виконувати певний аналіз представлених даних без знання про реальні результати, а навчання ж з підкріпленням хоч явних прикладів своїх результатів не має, але отримує відповідь до кожного зробленого кроку у вигляді функції нагороди (чим більша нагорода, тим правильніший результат представила модель). Таким чином можна коротко представити різницю між типами навчання. В дипломній роботі буде використана модель машинного навчання з вчителем. Його задача полягає в тому, щоб з маркованих навчальних даних створити модель, яка доволяє робити прогнози про дані, які раніше не зустрічала [2]. Для нього ключовими складовими є:

— приклади вхідних даних

— приклади результатів

— спосіб оцінки якості.

Модель навчається трансформувати надані їй вхідні приклади у надані приклади відповідних результатів. Алгоритм віддає свої власні результати, в процесі чого рахується функція оцінки, наскільки вони близькі до реальних.

Для окремої задачі краще підходить той чи йнший алгоритм. Існує певний теоретичний розподіл алгоритмів відповідно до видів задачі, але він є узагальненим. У машинному навчанні усі ідеї і гіпотези доводять практичним методом, тож підбирати алгоритм теоретично неможливо. Для цього будується декілька різних початкових моделей, натреновуються і обирається найкраща. Аби зробити цей вибір і зіставити моделі, необхідно спочатку прийняти рішення щодо метрики вимірювання якості. Найпоширеніші метрики:

— accuracy або вірність визначається як частка правильних відповідей моделі від загального числа прикладів;

— precision або точність — це частка прикладів, що дійсно належать до певного класу, від прикладів, віднесених моделлю до цього класу;

— recall або повнота — це частка прикладів, віднесених моделлю до певного класу від прикладів, що дійсно належать до класу;

— f-міра — це гармонічне середнє між точністю і повнотою моделі.

Під час будь-якої перевірки якості, чи то для вибору найкращого алгоритма, чи то для спостереженням над тренуванням, чи то для кінцевої оцінки моделі, необхідно відрізати меншу частину прикладів з набору даних і виконувати перевірку тільки на цій відібраній частині — тестовій виборці. Детальніше про це буде описано в наступних розділах.

2.3. Поняття глибокого навчання

Глибоке навчання — це розділ машинного навчання, що взяв за основу підхід багатошарового представлення. Модель ділиться на певну кількість шарів (або ж рівней), яку і називають глибиною моделі. В сучасних

алгоритмах глибокого навчання у процесі тренування беруть участь десятки чи сотні шарів, що визначаються під впливом вхідних даних. Інші ж підходи (поверхневе навчання) машинного навчання використовують у тренуванні один-два шари, що і демонструє причину виокремлення глибокого навчання як розділа машинного навчання.

2.4. Нейронні мережі глибокого навчання

В глибокому навчання такі багатошарові представлення майже завжди навчаються з використанням моделей, що мають назву нейронні мережі. Хоч термін „нейронна мережа“ позичений з нейробіології і деякі базуючі ідеї позичені з науки про мозок, моделі глибокого навчання не представляють собою модель мозку [1]. Їх робота базується на таких об’єктах і складових:

— шарах

— наборі даних

— функції втрат

— оптиримізаторі.

Про перші два було вже згадано в розділі, тож необхідно визначити інші два поняття.

Функція втрат або цільова функція, як її також називають, розраховує деяку кількісну оцінку успішності роботи мережі шарів. Ця оцінка повинна мінімізуватися в подальшому навчанні моделі. Якщо нейронна мережа має, наприклад, два виходи, тобто повинна повернути два різних значення (наприклад, при ідентифікації за віком і статтю), то вона матиме дві різні функції втрат, адже природа і діапазон цих двох відповідей відрізняються. Проте усі значення втрат необхідно поєднати (усереднити), що обумовлено алгоритмом навчання.

Оптимізатор, в свою чергу, визначає, як саме буде змінюватись мережа шарів під впливом функції втрат. Тобто за допомогою оптимізатора

розробник вказує, яким чином проходитиме навчання моделі.

Модель глибокого навчання представляє собою орієнтовний, ациклічний граф шарів. Шар — це модуль обробки даних — значень, які мережа отримає на вході, і які вона повинна привести до певного вигляду на виході. Спосіб, яким модель перетворить вхідні дані на бажані відповіді, якраз і визначається мережею шарів. Кожен з шарів має свій масив значень, результуючих в процесі своєї обробки, який називають вагами шара.

На Рисунку 2.3 зображена схема, що демонструє зв’зок усіх складових нейронної мережі: шари мережі об’єднуються в ланцюжок, і перетворюють вхідні дані у прогнози моделі. Функція втрат отримує їх і вірні значення, порівнює між собою і повертає оцінку, наскільки близькі результати моделі до бажаних за значенням. Оптимізатор використовує це значення, щоб змінити ваги шарів мережі.

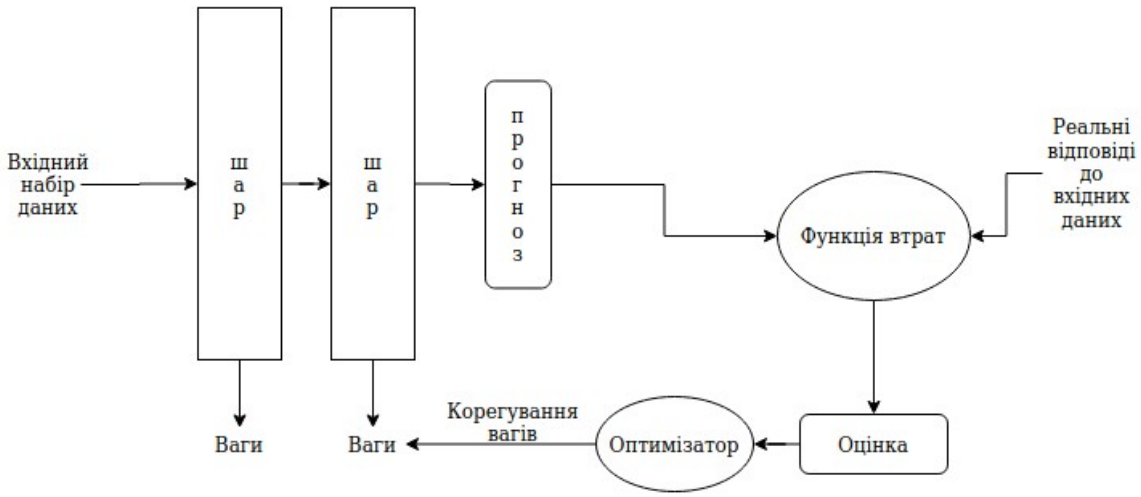


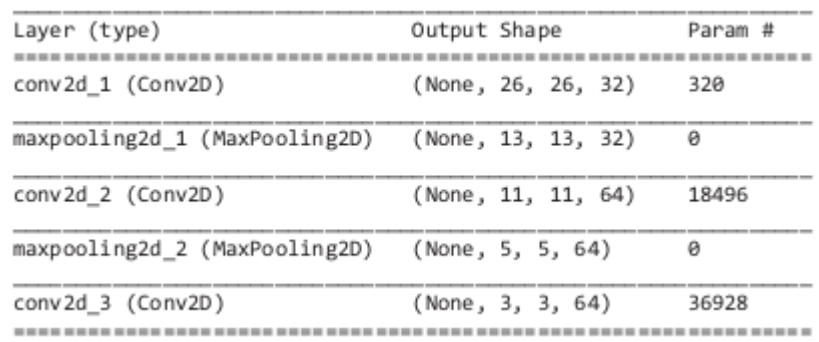
Рисунок 2.3 — Зв’язок складових нейронної мережі

2.5. Згорткові нейронні мережі

Згорткові нейронні мережі (Convolution Neural Network) є різновидом моделей глибокого навчання, майже повсякчас використовуваних в додатках

комп'ютерного зору (розпізнавання образів).

Приклад архітектури простої мережі класу Convolution Neural Network представлено на Рисунку 2.4.



Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 26, 26, 32)	320
maxpooling2d_1 (MaxPooling2D)	(None, 13, 13, 32)	0
conv2d_2 (Conv2D)	(None, 11, 11, 64)	18496
maxpooling2d_2 (MaxPooling2D)	(None, 5, 5, 64)	0
conv2d_3 (Conv2D)	(None, 3, 3, 64)	36928

Рисунок 2.4 — Приклад архітектури згорткової мережі

При використанні такої моделі для класифікації зображень рукописних цифр з набора даних з назвою MNIST, оцінка за метрикою accuracy становить 99.3%, що є надзвичайно хорошим результатом і демонструє ефективність згорткових мереж.

2.5.1. Операція згортки

Головна відмінність згорткової нейронної мережі і причина зростання ефективності ховається у назві — це згорткові шари. Особливість операції згортки у вивченні локальних шаблонів з даних (для зображення кішки, наприклад, шаблоном є частина зображення з її вухом), що наділяє мережу важливими властивостіми:

— після вивчення певного шаблону на зображенні, модель зможе знаходити його в інших прикладах, навіть якщо розміщення шаблону відрізнятиметься

— вивчення шаблонів відбувається у певній ієрархії, тобто спочатку вивчаються більш локальні з них, далі вони об’єднуються у більші і т. д. Так вивчення складних візуальних представлень проходить ефективніше.

2.5.2. Проблема перенавчання

У машинному навчанні типовою проблемою є перенавчання моделі.

Перенавчання проявляється в тому, що під час тренування модель демонструє високу оцінку якості, тоді ж при перевірці якості на тестовому наборі це значення значно менше. Це говорить про підлаштування моделі до даних, на яких вона натренована. Модель вивчила природу даних настільки добре, що будь-яке відхилення від цієї природи, яке вона бачить на тестових даних, вже заважає виконувати ефективно поставлену задачу. Зворотнім поняттям до перенавчання є недонавчання — коли модель не здатна навчитися розпізнавати образи (або ж залежно від задачі). Особливо схильні до перенавчання моделі глибокого навчання. Складність нейронних мереж, що викликана великою кількістю параметрів, зменшує їх якість на даних, що зустрічаються вперше.

2.5.3. Операція max-pooling

Шар з назвою max-pooling (вибір максимального значення з сусідніх) зменшує кількість параметрів, перебільшення яких призводило б до інтенсивного перенавчання у згорткових мережах. Це робиться з використанням операції вибору максимального значення. Аналогічно може реалізовуватись операція вибору середнього значення у шарі Average pooling. Але операція max pooling на практиці переважає ефективністю.

2.6. Приклади вирішення задачі

На даному етапі розвитку машинного навчання, з’являється все більше систем штучного інтелекту з різноманітнішими призначеннями. Не тільки науковці, а і студенти з технологічних спеціальностей, люди, що навчаються програмуванню або діючі працівники сфери машинного навчання реалізують свої ідеї і можливості, розроблюючи системи, що прийматимуть певні рішення, такі як класифікація відео та зображень. Багато рішень на розглянуту тему представлено та описано в відкритому доступі. Але оскільки більшість з них є продуктами з метою дослідження, вони мають обмежений

функціонал і майже відсутній інтерфейс. Аналогів програмному продукту, що розроблений в межах дипломної роботи, нема. Більші повноцінні системи, що в основному є комерційними, не представляють функціонал, повністю відповідний даному.

Проект Age Gender Prediction доступний на ресурсі github.com. Розробник пропонує завантажити файли з програмним кодом і запустити їх, вказавши свій файл для обробки і у результаті можна отримати висновок про класифікацію людей на зображенні. Обличчя людини діляється і вказується нижче її вік і стать (рисунок 2.1). Такий функціонал є достатньо простим, а результат — зрозумілим, але обмеженим і недостатньо зручним у використанні. Додатку не вистачає користувацього інтерфейсу. Якість роботи системи оцінена метрикою accuracy на 90.18%.

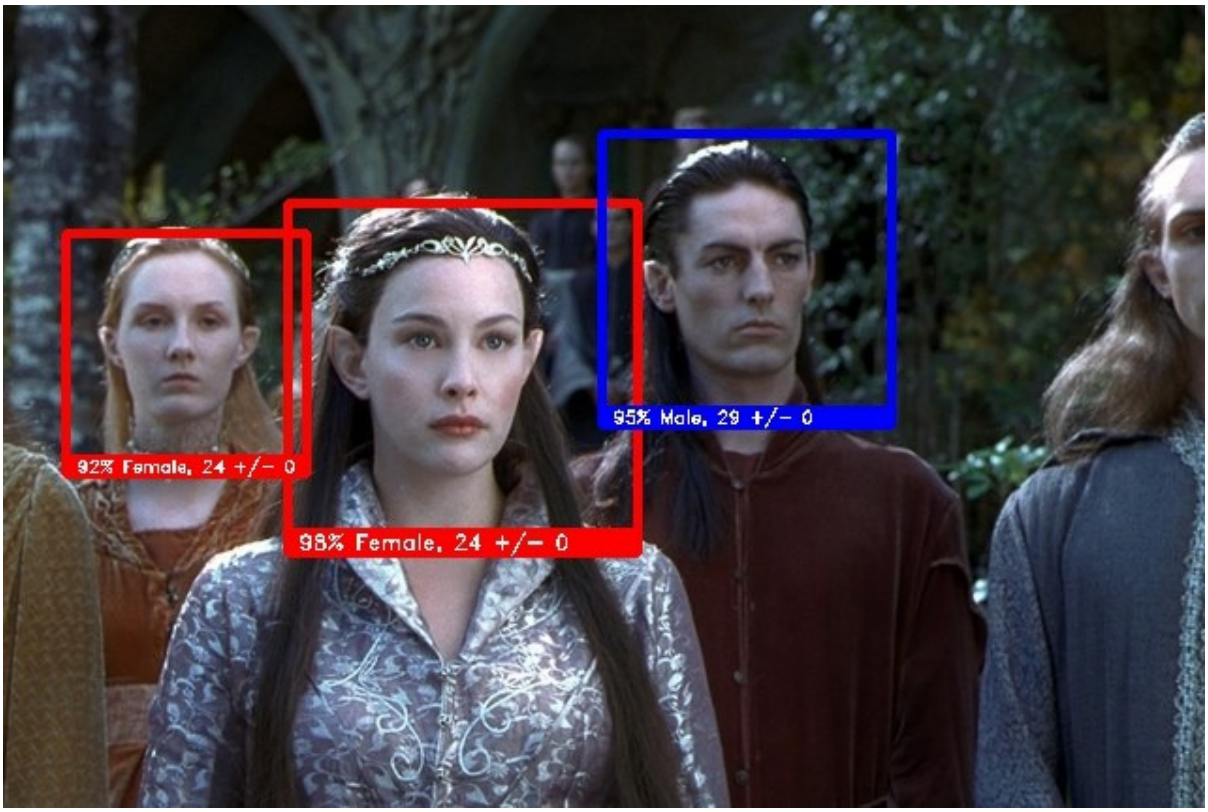


Рисунок 2.5 — результат роботи системи Age Gender Prediction

Стаття під назвою “Predicting customer’s gender and age depending on

mobile phone data” [3] описує додаток, що прогнозує стать і вік користувачів на основі різної інформації про них, пов’язаної з поведінкою телекомунікаційним зв’язком. Для навчання і тестування моделі компанія “СіріяТелТелеком” надала набір даних про 18 000 користувачів. Модель, розроблена з використанням технологій Big Data, досягла 85.6% якості accuracy для прогнозування статевої групи і 65.5% для прогнозувавання віку. Дана система має доволі низькі показники якості і потребує велику кількість даних для прогнозування, тоді як отримувати прогноз на основі одної фотографії або відео набагато доступніше з точки зору збору даних. Описана система, натомість, може бути корисною в сфері свого використання і мати високий показник якості порівняно з аналогічними рішеннями.

2.7. Висновки до розділу

В цьому розділі було описано сучасні методи вирішення задачі ідентифікації людей, які представляють собою алгоритми машинного навчання. Були визначені поняття штучного інтелекту, машинного навчання та глибокого навчання, встановлено чим вони відрізняються і яким чином пов’язані, описано поняття нейронної мережі і її складових. Визначено поняття згорткової нейронної мережі та причини її переваг у вирішеннях задачі розпізнавання образів.

Як теоретичний приклад сучасного методу вирішення задач машинного навчання було представлено приклад згорткової нейронної мережі для класифікації рукописних чисел з розбором поняття згорткової нейронної мережі. Як практичні приклади вирішення задач, подібних до поставленої, було описано дві подібні до розробленої в межах дипломної роботи системи з порівнянням якості і обґрунтуванням переваг і недоліків кожної.

3. ЗАСОБИ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1. Мова програмування Python

Python — інтерпретована об’єктно-орієнтована мова програмування високого рівня з динамічною семантикою. Інтерпретатор Python і велика стандартна бібліотека є доступними і безкоштовними для всіх основних платформ, таких як Windows, Linux і MacOS. Аудиторія користувачів Python складає близько одного мільйона. Виділені основні переваги цієї мови програмування: якість програмного забезпечення, продуктивність розробника, портативність програм, підтримка бібліотек, інтеграція компонентів, легкість роботи з мовою.

3.1.1. Якість програмного забезпечення

Відрізняють Python від інших інструментів його простий і читаємий синтаксис, узгодженість та якість програмного забезпечення в цілому. Оскільки код на Python розроблений так, що його легко читати, то ще й багаторазово використовувати і обслуговувати. Усі особливості мови основані на невеликому наборі базових понять, тому взаємодіють узгоджено і обмежено. Це робить сову програмування легкою до вивчення, до запам’ятовування основних концепцій і розуміння. Python підтримує модулі, об’єктно-орієнтоване програмування і функціональне програмування, що являються інструментами для повторного використання програмного забезпечення.

3.1.2. Продуктивність розробника

Продуктивність розробки на мові Python значно обходить скомпільовані або статично набрані мови, як C, C++, Java. Розробники, що використовують Python, економлять час друку, налагодження і підтримки коду. Наприклад, розміри еквівалентних шматків кода на Python та C++ або

Java у 3-5 разів відрізняються розмірами. Швидкість розробки забезпечується також високою швидкістю запуску кода без довгих етапів компіляції і компонування.

Отже, мова програмування Python спеціально оптимізована під швидкість розробки — вона гарантує простий ситаксис, динамічний набір тексту, відсутність етапів компіляції та вбудований інструментарій [4].

3.1.3. Портативність програм

Більшість програм, написаних мовоб програмування Python, працюють без змін на усіх основних комп’ютерних платформах [4]. Тобто достатньо лише скопіювати програмний код між машинами різних операційних систем. Портативність Python пропонується для графічних інтрфейсів користувача, програм доступа до баз даних, веб-систем, і навіть інтерфейсів операційних систем.

3.1.4. Підтримка бібліотек

Стандартна бібліотека Python — це величезна колекція готових функцій, що підтримує багато задач програмування. Окрім того, Python може бути доповнений програмним забезпеченням сторонніх розробників для підтримки додатків. Деякі з них будуть описані пізніше у розділі.

3.1.5. Інтеграція компонентів

Скрипт Python може легко взаємодіяти з іншими частинами додатку. Для цього використовується механізми інтеграції, що дозволяють виокристовувати Python як інструмент налагодження та розширення програмного продукта.

3.1.6. Легкість роботи з мовою

Легкість у використанні, розумінні та багато налагоджених і узгоджених інструментів мови гарантують більшості розробникам задоволення від роботи з Python. Це є важлива перевага для питання продуктивності роботи програміста, адже перетворює програмування з рутинної роботи на цікаву і захоплюючу діяльність з безліччю можливостей.

3.1.7. Використання Python для машинного навчання

Python — одна з найбільш популярних мов програмування, що використовуються для роботи з машинним навчанням і причинами цьому є вже раніше описані її переваги. Товариство програмістів розробило і продовжує поповнювати величезну кількість корисних бібліотек з відкритим вихідним кодом для виконання задач програмування машинного навчання.

Оскільки продуктивність мов програмування по типу Python гірша для обчислювально громістких задач, ніж у мовах нижчого рівня, для векторизованих операцій з масивами, в тому числі і багатовимірними, були розроблені бібліотеки NumPy і SciPy, що спираються на низькорівневі реалізації на Fortran і C.

3.2. Бібліотека глибокого навчання Keras

Keras — це фреймворк підтримки глибокого навчання для мови програмування Python, що забезпечує зручний спосіб створення і тренування майже будь-яких моделей глибокого навчання. Спочатку Keras розроблявся для гарантування швидкого проведення експериментів під час досліджень.

Ключові характеристики фреймворка наступні:

— дозволяє запускати код на CPU або GPU без суттєвих змін;

— API спрощує розробку прототипів моделей;

— підтримка згорткових мереж та рекурентних мереж, а також різноманітних комбінацій;

— підтримка довільних мережевих архітектур: моделі, що мають декілька входів або виходів, спільне використання шарів або моделей. Усе це гарантує можливість створення приктично будь-яких моделей глибокого навчання з допомогою Keras.

Також, фреймворк розповсюджується з умовами вільної ліцензії і безкоштовно використовується в комерційних проектах [1]. Такі відомі

компанії, як Google, Netflix, Uber, CERN, Yelp, Square, використовують Keras.

3.3. Бібліотека OpenCV

OpenCV — це бібліотека з відкритим вихідним кодом для розробки додатків комп’ютерного зору, які можливо запускати на декількох платформах, таких як Windows, Linux, Mac, Android і iOS. Ліцензія BSD дозволяє вільно використовувати, розповсюджувати і адаптувати OpenCV.

Бібліотека пропонує вирішення найбільш фундаментальних задач з обробки зображень, таких як читання, відображення, збереження. Також OpenCV використовує структуру даних для обробки зображень і матриць — ця структура є потужною і має багато корисних атрибутів і методів. Значно спрощує роботу сучасна модель управління пам’яттю [5].

3.4. СУБД MySQL

Система баз даних MySQL використовує архітектуру клієнт-сервер, яка зосереджується навколо сервера, mysqld. Сервер — це програма, яка фактично маніпулює базами даних. Клієнтські програми не роблять це безпосередньо; швидше, вони повідомляють ваш намір серверу за допомогою запитів написаних структурованою мовою запитів (SQL). Клієнтська програма, або програми, встановлені локально на машині, з якої ви хочете отримати доступ до MySQL, але сервер можна встановити в будь-якому місці, поки клієнти можуть підключитися до нього. MySQL — це невід'ємна мережева система баз даних, тож клієнти можуть спілкуватися з сервером, який працює локально на вашій машині або тим, який працює деінде, можливо, на машині з іншого боку планети.

Клієнти можуть бути написані для різних цілей, але кожен взаємодіє з

сервером підключившись, відправляючи туди запити SQL для виконання операцій з базою даних та отримаючи від нього результати запиту [6].

У SQL команда JOIN використовується для порівняння, об`єднання та повернення конкретних рядків даних з двох або більше таблиць у базі даних. На рисунку 3.1 продемонстрована різниця між типами JOIN, де відношення представлені у вигляді кіл.

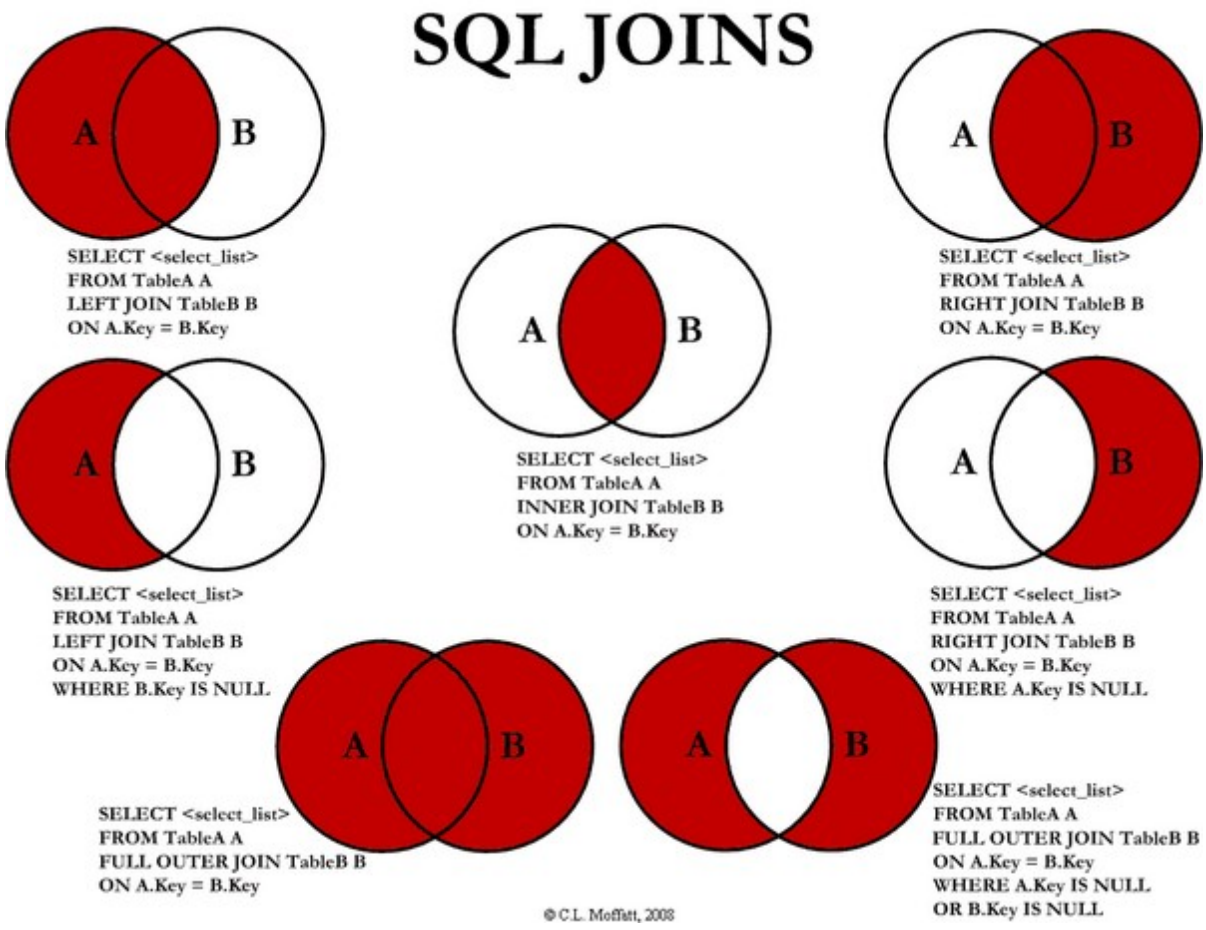


Рисунок 3.1 — Типи SQL команди JOIN

INNER JOIN знаходить і повертає відповідні дані з таблиць, тоді як OUTER JOIN знаходить і повертає відповідні дані та деякі різні дані з таблиць. Існує три типи OUTER JOIN: LEFT OUTER JOIN, RIGHT OUTER JOIN та FULL OUTER JOIN.

3.5. Бібліотека для створення інтерфейсу Kivy

Інтерфейс додатку був розроблений з використанням Kivy — бібліотеки з відкритим вихідним кодом для створення кросплатформних UI додатків і ігр на мові програмування python. Kivy працює на платформах Linux, Windows, OS X, Android, iOS та Raspberry Pi. Kivy може в основному використовувати більшість входів, протоколів та пристроїв, включаючи WM_Touch, WM_Pen, Mac OS X Trackpad та Magic Mouse, Linux Kernel HID. У комплект входить симулятор миші з декількома дотиками. Графічний двигун побудований над OpenGL ES 2.

Окрім визначення конструкції дерев віджетів з чітким оголошенням прив’язок з допомогою програмного коду, бібліотека підтримує використання мови kv. Мова kv, яку називають kvlang або kivy language, дозволяє створювати дерево віджетів декларативним способом і зв'язувати властивості віджетів один з одним або природним чином здійснювати зворотний зв'язок. Мова Kivy полегшує розділення логіки та представлення програми. Це фундаментальна інженерна концепція, що допомагає зберегти простий та інтуїтивно зрозумілий код [6]. Тим не менш, додаток Kivy можна побудувати, використовуючи чистий Python з імпортованою бібліотекою Kivy.

3.6. Unit тестування

Концепція автономного (або Unit) тестування зародилася ще в 1970-х роках при створенні мови програмування Smalltalk. Ще з того часу вона залишається один з найкращих способів покращення кода і дозволяє розробнику глибше розуміти функціональні вимоги до класа або методу [8].

Для розуміння поняття автономного тесту необхідно визначити, з

якими елементами коду чи програми він повинен працювати. Існує окреме поняття для їх визначення — одиниці роботи. Одиниця роботи — це сукупність певних операції, що починається з момента виклику певного метода і закінчується при отриманні тестом кінцевого результату.

Якісний автономний тест має певні характеристики і властивості:

— простий у реалізації;

— має можливість швидкого запуску (наприклад, нажаттям однієї кнопки);

— швидкий у виконанні;

— автоматизований і повторюваний;

— зберігає свою актуальність у будь-який момент існування системи;

— має стаільні результати виконання;

— повністю контролює одиницю, яку тестує;

— ізольований і незажений від виконання інших тестів;

— при невдалому виконанні надає зрозумілу інформацію про зміст і місце помилки.

Отже, враховуючи усе вище сказане, можна надати визначення поняттю автономного тестування: автоматизована частина програмного коду, що викликає одиницю роботи, яку необхіно протестувати, і виконує перевірку певних припущень про кінцевий результат цієї одиниці [8].

3.7. Docker контейнери

Контейнери сильно впливають на спосіб розробки, розповсюдження і функціонування програмного забезпечення. Вони дають змогу розробникам створювати своє ПЗ на локальній системі з гарантією його подальшої незмінної роботи в будь-якій операційній системі.

Контейнер — це засіб інкапсуляції додатка разом з його залежностями. Іноді контейнери порівнюють з віртуальними машинами, проте у контейнерів

є певні суттєві переваги. Контейнери використовують ресурси основної операційної системи, що робить їх більш ефективними. Контейнери можна запускати або зупиняти за долі секунди. Також спрощена сутність контейнера означає, що розробники можуть запускати одночасно декілька контейнерів. Це дає можливість імітувати роботу розподіленої системи.

Найважливішим є те, що метою використання віртуальної машини є повна емуляція чужерідної операційного або програмного середовища, а мета контейнера — зробити додаток самодостатнім і можливим до переносу.

Основна мета Docker — перенести переваги стандартизації контейнерів у область інформаційних технологій. Контейнери Docker значно спрощують переміщення програм.

Платформа Docker складається з двох компонентів: Docker Engine і Docker Hub. Перший представляє собою механізм, щовідповідає за створення і функціонування контейнерів, а другий є хмаровим сервісом для розповсюдження контейнерів. В основу Docker вкладена технологія Linux-контейнерів з певними обгортками і розширеннями [9].

3.8. Висновки до розділу

В цьому розділі було описано усі засоби розробки програмного забезпечення за темою дипломної роботи: “Аналіз відеопотоку: ідентифікація людей за статтю і віковою групою”. Серед них мова програмування Python, щодо якої описано усі найголовніші переваги як сучасної поширеної серед розробників мови програмування. Описані бібліеотеки Python, такі як Keras — бібліотека глибокого навчання, OpenCV — бібліотека для додатків комп’ютерного зору та Kivy — бібліотека створення графічного інтерфейсу користувача. Також серед засобів розробки є субд MySQl для реалізації бази даних системи. У розділі представлено визначення та головні переваги Unit тестування та контейнерів.

4. ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

Розроблена система має наступні компоненти, процес створення і реалізація яких описані в даному розділі: натренована модель машинного навчання, база даних, інтерфейс системи.

Процес створення системи ідентифікації людей за статтю та віковою групою передбачає умовно насамперед два основних етапа: створення і навчання моделі машинного навчання, створення додатку. Хоча програмний продукт, який бачитиме користувач, реалізується у другому етапі, що містить значно більшу частину програмного коду, перший етап не менш важливий.

Модель є основою системи, її ядром. Програмний код для зчитування, обробки і підготовки даних, для створення моделі та для її тренування і перевірки результатів, написаний на етапі створення моделі, буде запущений один раз. Його результатом буде деякий масив чисел, який надалі можна використовувати в програмному продукті. Дана частина роботи визначається не громісткістю, а складністю, тому обидва етапи співпадають по тривалості виконання.

4.1. Створення моделі машинного навчання

Розробка системи машинного навчання, що використовуватиметься для прогнозування, типово охоплює декілька ключових етапів:

— початкова обробка набору даних;

— тренування моделі;

— оцінка роботи моделі;

— прогнозування нових даних з допомогою створеної моделі.

Усі вище перераховані етапи, також їх складові та проміжні етапи описані схематично на Рисунку 4.1.

Особливо важливим етапом розробки модела машинного навчання є робота з даними. Якість прогнозуючої моделі дуже сильно залежить від вміння розробника процювати з даними, обробляти набори даних, аналізувати їх. Будь-яка необачність під час реалізації етапу підготовки і обробки даних, або ж неосвідченість розробника щодо науки про дані, призведе до неможливості натренувати модель. Якою б якісною за своєю архітектурою вона б не була, помилки з набором даних сильно впливають на якість, призводять до перенавчання або недонавчання, але найголовніше — зрозуміти їх причини важко, не розуміючи розподілу і формату своїх даних. Тож до даного етапу розробникам часто доводиться повертатися знову і знову при виникненні проблем на етапі тренування, адже дані- це ключ до тренування якісної і робочої моделі машинного навчання.

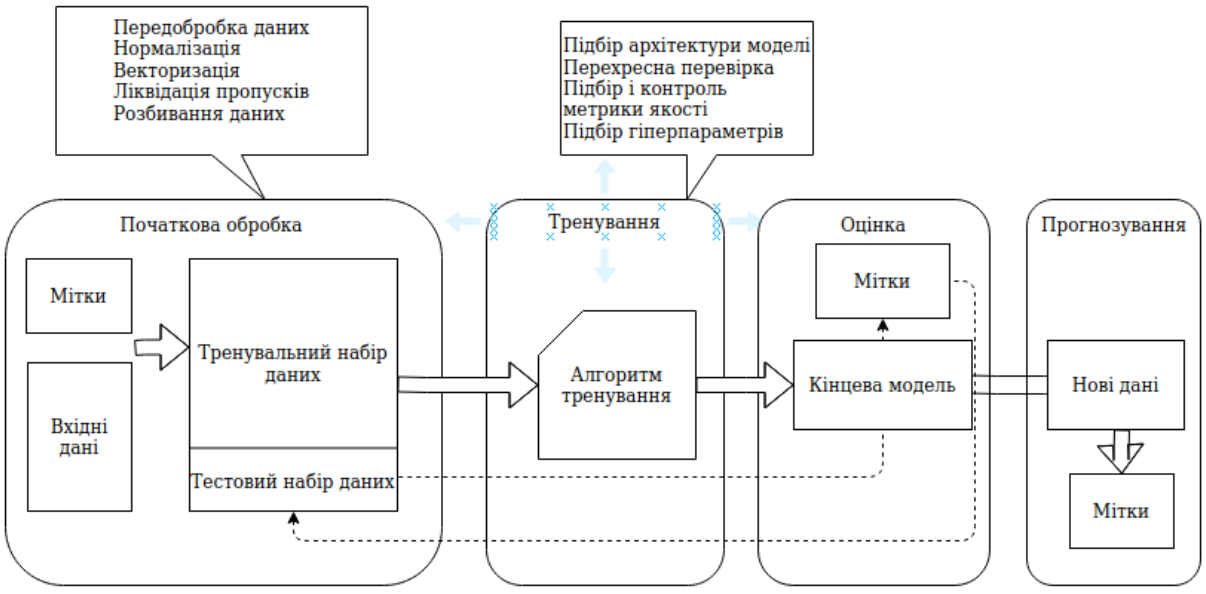


Рисунок 4.1 — Етапи створення моделі машинного навчання

4.1.1. Підготовка набору даних

Існує декілька обов’язкових для виконання пунктів для обробки набору даних на шляху до отримання найбільш якісного датасету, таких як:

— вирішення проблеми пропущених даних;

— кодування міток;

— обробка даних для тренування нейронної мережі;

— розділення набору на виборки.

Часто зустрічаються випадки, коли деяких або навіть багатьох екземплярах з даних зустрічаються пропущені значення. Причинами цього можуть бути: помилка про автоматизованому зборі даних, незаповнені поля в результаті опитування, недопустимі значення в наборі тощо. Усі компанії, що пропонують розробникам зібрані дані, роблять усе можливе для уникнення цієї проблеми, але варто пам’ятати, що дані, використовувані у створенні ефективних моделей машинного навчання, є виключно реальними, представленими людьми, тож природньо вони матимуть певні помилки, невідповідності і пропуски. Ресурси даних не обробляють пропуски самостійно і все одно залишають у пропонованих наборах екземпляри з одним-двома пропущенимим значеннями аби надати розробнику можливість вирішити це питання тим способом, який буде найефективнішим для його системи. Також, навіть екземпляри з пропущеними даними можуть нести в собі необхідну інформацію.

Найпростіший спосіб вирішення проблеми пропущених даних - це звичайне видалення екземплярів або признаків, в яких містяться пропуски. При виборі цього методу обов’язкоко пам’ятати, що недопустимо видаляти значну частину даних, адже можливо втратити певну інформацію в даних.Разом з видаленням пропущених полів завжди доводиться видаляти поля, що містять дані. Видалення проводиться або по стовпцям (признаки), або по рядкам (екземплярам).

Видаляючи екземпляри з пропущеними даними порібно враховувати загальний об’єм вибірки, величину частки пропущених значень та розподіл признаків. При малих об’ємах використовуваної вибірки небажано боротися з пропущеними значенням саме шляхом видалення. Причиною небезпечного

видалення даних також може бути зміна розподілу деяких признаків після видалення. Це питання потрібно контролювати в процесі, також можливо, ця зміна розподілу несе у собі певну інформацію і розробник зможе виділити окремий признак з відомостей про пропущені значення.

Особливо обережними необхідно бути при видаленні цілого признака з пропущеними даними. Якщо він несе цінність для майбутньої моделі, невелика кількість пропусків не є причиною відмови від неї.

Окрім видалення стовпців чи рядків, є інші способи боротьби з проблемою пропусків — це різноманітні способи інтерполяції [2]. Один з них — замінити пропущене поле середнім арифметичним значенням ознакового стопчика.

Після того, як пропущені дані і мітки ліквідованого, коректність даних перевірена, необхідно звернути особливу увагу на мітки. Більшість бібліотек машинного навчання вимагають від розробника представлення міток у цілочисельнову вигляді. Якщо мітки класа “Стать”, наприклад, мають значення типу “Female” та “Male”, то їх в такому випадку потрібно кодувати. І не є суттєво важливим, якими цілочисельними значеннями класи будуть закодовані, але популярною практикою є використання нумерування класів у будь-якому порядку, починаючи з нуля.

Так само, як і мітки, самі дані побрібно обробити, привівши їх формат і значення до вигляду, з яким модель зможе тренуватися. Якщо говорити про дані у вигляді зображень, то варто відмитити, що будь-який тип даних необхідно векторизувати. Тож зображення мають бути представлені масивами чисел. Полегшує тренування моделі деякі ознаки розподілу даних, наприклад однорідність — тобто усі признаки мають вміщатися приблизно в один діапазон значень. Досягнути цього можливо виконанням нормалізації. В деяких випадках нормалізація не тільки корисна, а й необхідна. Нормалізація даних формату векторизованих зображень (масивів пікселів зображення) відбувається шляхом віднімання середнього значення з кожного пікселя і

діленням його на стандартне відхилення. До процесу обробки даних-зображень також входять зміна розмірності зображень та зміни кута нахилу. Модель очікує на вхід масиви певного розміру і варто враховувати, що пронози моделі будуватимуться не на основі зображень високої якості, частіше зображення з камер матимуть низьку якість і модель має бути до цього пристосована. Аналогічна причина додавання зображень різного кута нахилу — поворот зображення на певний кут збільшить пристосованість моделі до різноманітних реальних даних.

Для тренування моделі було обрано набір даних для оцінки віку та статі людей на основі зображення під назвою IMDB-WIKI. IMDB-WIKI — це найбільший доступний набір даних для оцінки віку та статі людей, що містить більш ніж півмільйона зображень з мітками. Більшість відомих датасетів з зображеннями обличчя людей на даний момент часу або надто малі, або містять тільки фронтально вирівняні фотографії, або мають надто багато пропусків у мітках віку [10]. Для створення датасету IMDB-WIKI було використано зображення знаменитостей (Рисунок 4.2) з ресурсів IMDb та Wikipedia. Для цього проходячи список зі 100 000 найпопулярніших акторів, що перераховані на сайті IMDb, автоматично переглядалися їх профілі з усіма необхідними даними для розмітки і усіма фотографіями, що пов’язані з людиною. Після цього переглядалися всі зображення зі сторінки Wikipedia з відповідною інформацією.

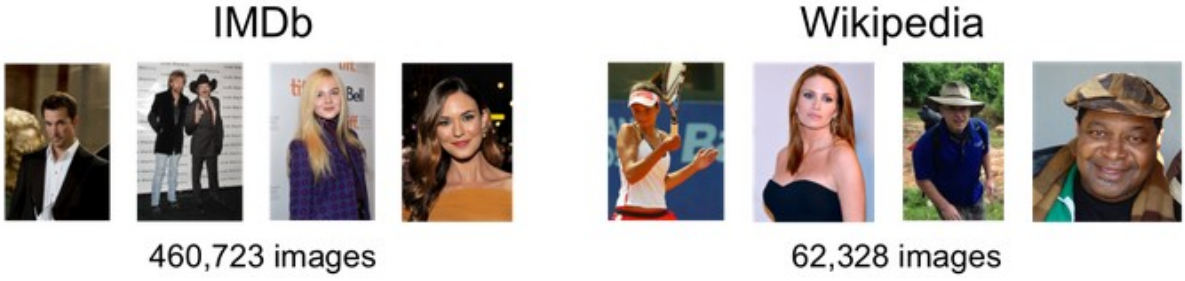


Рисунок 4.2 — Приклади зображень з набору даних IMDB-WIKI

Усі дані про кожну знаменитість та усі фотографії оброблялися і

збиралися якісно аби забезпечити коректність даних, відсутність пропущених даних та міток. Для цього необхідно було обійти деякі нюанси і недоліки в даних з ресурсів IMDb та Wikipedia, наприклад: деякі зображення з сайту IMDb містили декілька людей на фотографії, тож фотографії відбиралися з використанням технології розпізнавання обличчя, або деякі зображення з Wikipedia не містили рік, що не дає можливості правильно вказати вік людини на фото.

Для забезпечення якісного тренування моделі віковий розподіл у наборі був вирівняний, що значно полегшило для розробників, що використовують цей датасет, процес обробки даних. Розподіл віку в наборі даних IMDB-WIKI та окремо в двох складових цього набору зображено за допомогою графіка на Рисунку 4.3.

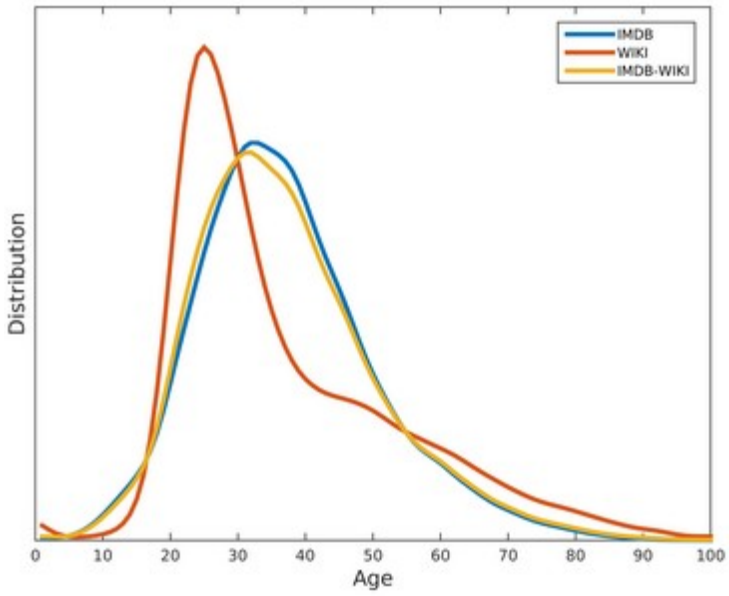


Рисунок 4.3 — Розподіл віку у наборі даних IMDB-WIKI

Оскільки на якість натренованої моделі сильно впливає величина об’єму набора даних, необхідно обирати найбільші з доступних датасетів. Усього набір даних IMDB-WIKI містить 523 051 зображень обличчя, серед яких 460 723 зображень 20 284 знаменитостей з IMDb та 62 328 узятих з Wikipedia. Лише 5% знаменитостей мають більше 100 фотографій, а в

середньому кожна знаменитість має 23 фотографії [10].

4.1.2. Створення архітектури моделі

Як було зазначено у попередніх розділах, у розпізнаванні зображень найвищу якість демонструють згорткові нейронні мережі. За основу цього класу мереж вкладено операцію згортки, яка виконується в згортковому шарі. Згортка представляє собою операцію між двома масивами чисел, в результаті якої за певною функцією один масив (зображень в даному випадку) фільтрується іншим. Функції фільтрації представляються невеликими матрицями — ядрами. На Рисунку 4.4 видно, як до певного елемента зображення застосовується ядро 3*3. Подібна операція проводиться для усіх елементів зображення, отримуючи в результаті еквівалентне зображення, на якому будуть виділені одні елементи (певні лінії, деталі, для перших шарів мережі це скоріше за все будуть края зображення), а інші розмиті.

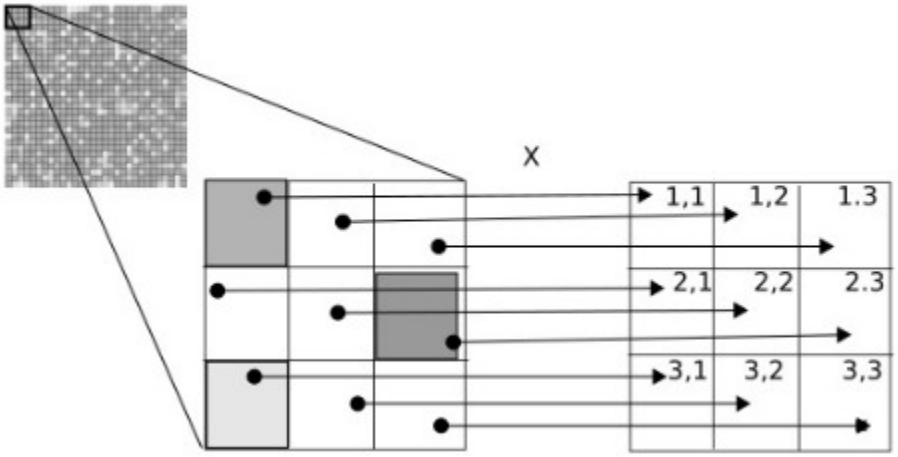


Рисунок 4.4 — Операція згортки

Ядро згортки має додаткові параметри, що впливають на виконання операції — це крок і відступ. Крок — це кількість одиниць переміщення для ядра [11]. На Рисунку 4.5 представлено кілька варіантів кроку для переміщення, серед яких третій випадок не співпадає по розміру, адже неможливо виконати останній крок ядра.

Чим більше ядро, тим більше одиниць на границі матриці не отримають відповіді, адже необхідно охопити ядро повністю. Тому до зображення додається границя заданої ширини — відступ, щоб ядро рівномірно розподілити по краям [11].

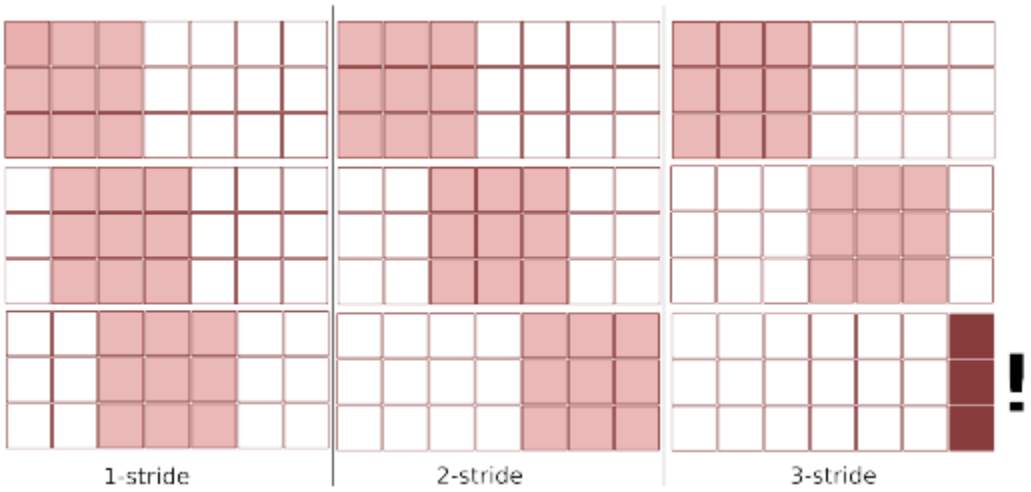


Рисунок 4.5 — Варіанти кроку ядра

Pooling layer відповідає за зменшення просторових розмірів згорткового елемента. Це робиться для зменшення обчислювальної потужності, необхідної для обробки даних, за рахунок зменшення розмірності. Крім того, він корисний для вилучення домінуючих характеристик, таким чином підтримуючи процес ефективного навчання моделі. Існують різні типи об'єднання. Один з них, Max Pooling, повертає максимальне значення з частини зображення, покритої ядром. Приклад цієї операції представлено на Рисунку 4.6: для кожного polling-ядра розміром 2*2 матриці 16*16 проводиться вибір максимального елемента і в результаті отримується матриця 4*4. Для цієї операції вибрано функцію вибору максимального числа, тоді як можна так само застосувати функцію розрахунку середнього значення.

У нейронної мережі, функція активації відповідає за перетворення сумарного зваженого входу з вузла в активацію вузла або вихід для цього

входу. Функція корекції лінійної активації (ReLU) є кусково-лінійною функцією, яка буде виводити вхід безпосередньо, якщо він позитивний, в іншому випадку, вона буде виводити нуль. Вона стала функцією активації за замовчуванням для багатьох типів нейронних мереж, оскільки модель, яка використовує її, легше тренується і часто досягає кращих результатів.

Як було описано в минулих розділах, часто в глибокому навчанні зустрічається проблема перенавчання. В нейронних мережах перший крок до її вирішення — додавання шару з назвою Dropout. Його мета — звести до нуля випадкові значення вагів, кількість яких обирається параметром шара. Таким чином мережа ніби частково втрачає знання, що отримала в процесі попереднього тренування, і продовжує вчитися далі.

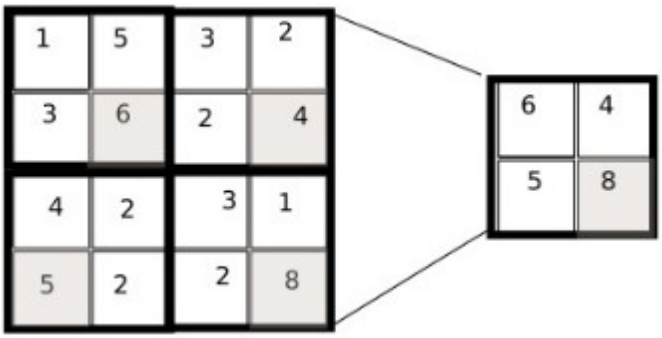


Рисунок 4.6 — Max Pooling з ядром розміру 2*2 для матриці 16*16

Додання ефективності тренуванню моделі за рахунок збільшення швидкості навчання можливо з допомогою технології пакетної нормалізації — batch normalization. Стандартний спосіб виконання цієї операції полягає в наступному: для кожного пакета даних віднімається середнє його вибірки і ділиться на її дисперсію, з допомогою чого отримується розподіл з центром в 0 і дисперсією 1, що пришвидшує процес тренування мережі.

З ростом глибини мережі процес тренування ускладнюється. Вирішення проблеми було запропоновано у грудні 2015 з появою нової архітектури залишкових мереж — Residual Networks (ResNet), ідея якої — поєднати вихід з блока шарів згортки разом з їх оригінальним входом.

Залишковий блок, що реалізує цю операцію зображено на Рисунку 4.7. Вона не додає додаткових параметрів і обчислювальної складності моделі, натомість такі мережі стало легше оптимізувати і збільшити точність зі збільшенням глибини [12].

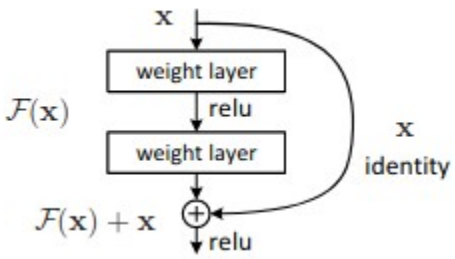


Рисунок 4.7 — Залишковий блок

Основна мета залишкових мереж — зробити мережу настільки тонкою, наскільки це можливо. Для виконання мети було створено “тонший” блок — BottleNeck [13], який використовує згортку з ядром 1*1 аби зменшити кількість каналів до виконання дорогої з точки зору обчислення згортки з ядром 3*3. Після цього виконується ще одна згортка 1*1, для повернення до початкової форми. На Рисунку 4.8 представлено приклади базового “BasicBlock” (справа) залишкової мережі і BottleNeck блока (зліва).

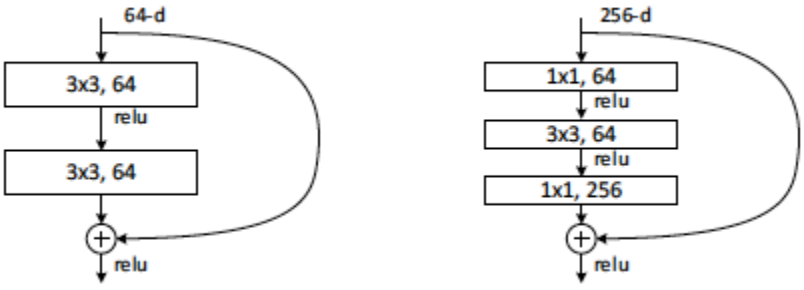


Рисунок 4.8 — Два типи блока залишкової мережі

Наступний крок до покращення архітектури моделі — це широкі залишкові мережі (Wide Residual Network), різниця яких зі звичайними

залишковими у розширенні моделі за рахунок збільшення числа каналів через додатковий параметр k (Рисунок 4.9).

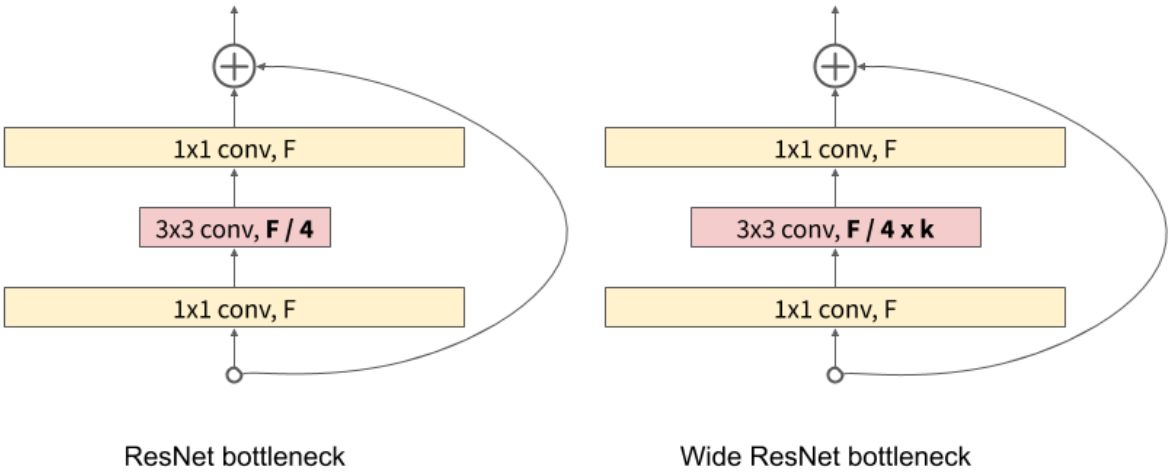


Рисунок 4.9 — Блок широкої залишкової мережі

Розширення блоків залишкових мереж — це більш ефективний спосіб підвищення ефективності роботи залишкових мереж порівняно зі збільшенням глибини. З обчислювальної точки зору це доводиться тим, що GPU набагато ефектиніше у паралельних обчисленнях на великих тензорах даних. Щодо практичного доведення, широка залишкова мережа на 16 шарів має рівну точність до точності 1000-шарової тонкої залишкової мережі, але в декілька разів перевищує її по швидкості навчання [13].

4.1.3. Опис архітектури

Для прогнозування віку і статі людини за допомогою обробки зображення була використана модель широкої залишкової нейронної мережі. Архітектура цієї мережі представлена на Рисунках 4.10-4.12 у вигляді таблиці. Кожен рядочок відповідає шару мережі. Таблиця має чотири стовпчики. Перший позначає тип шару, серед можливих типів:

— вхідний шар — InputLayer;

— згортковий — Conv2D;

— шар активації ReLU — Activation;

— шар пакетної нормалізації — Batch Normalization;

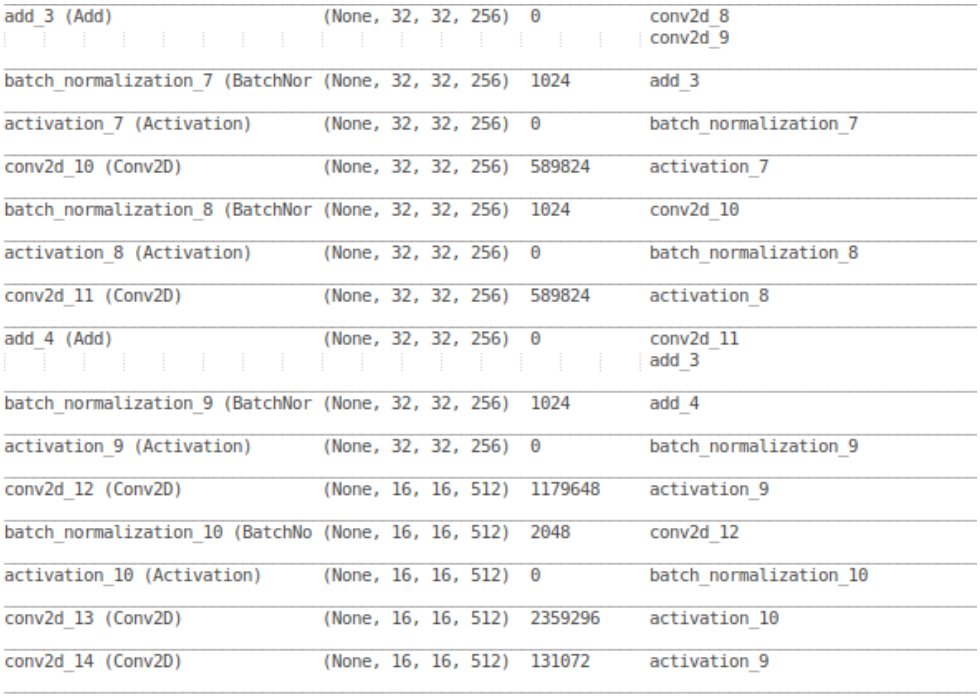
— функція додавання з двома входами з інших шарів — Add — для реалізації блоку залишкової мережі.

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	(None, 64, 64, 3)	0	
conv2d_1 (Conv2D)	(None, 64, 64, 16)	432	input_1
batch_normalization_1 (BatchNor	(None, 64, 64, 16)	64	conv2d_1
activation_1 (Activation)	(None, 64, 64, 16)	0	batch_normalization_1
conv2d_2 (Conv2D)	(None, 64, 64, 128)	18432	activation_1
batch_normalization_2 (BatchNor	(None, 64, 64, 128)	512	conv2d_2
activation_2 (Activation)	(None, 64, 64, 128)	0	batch_normalization_2
conv2d_3 (Conv2D)	(None, 64, 64, 128)	147456	activation_2
conv2d_4 (Conv2D)	(None, 64, 64, 128)	2048	activation_1
add_1 (Add)	(None, 64, 64, 128)	0	conv2d_3 conv2d_4
batch_normalization_3 (BatchNor	(None, 64, 64, 128)	512	add_1
activation_3 (Activation)	(None, 64, 64, 128)	0	batch_normalization_3
conv2d_5 (Conv2D)	(None, 64, 64, 128)	147456	activation_3
batch_normalization_4 (BatchNor	(None, 64, 64, 128)	512	conv2d_5
activation_4 (Activation)	(None, 64, 64, 128)	0	batch_normalization_4
conv2d_6 (Conv2D)	(None, 64, 64, 128)	147456	activation_4
add_2 (Add)	(None, 64, 64, 128)	0	conv2d_6 add_1
batch_normalization_5 (BatchNor	(None, 64, 64, 128)	512	add_2
activation_5 (Activation)	(None, 64, 64, 128)	0	batch_normalization_5
conv2d_7 (Conv2D)	(None, 32, 32, 256)	294912	activation_5
batch_normalization_6 (BatchNor	(None, 32, 32, 256)	1024	conv2d_7
activation_6 (Activation)	(None, 32, 32, 256)	0	batch_normalization_6
conv2d_8 (Conv2D)	(None, 32, 32, 256)	589824	activation_6
conv2d_9 (Conv2D)	(None, 32, 32, 256)	32768	activation_5

Рисунок 4.10 — Архітектура моделі системи

Другий стовпчик містить розмірність вихідного масиву, третій — кількість параметрів. Четвертий стовпчик таблиці вказує на зв’язок шарів мережі, а

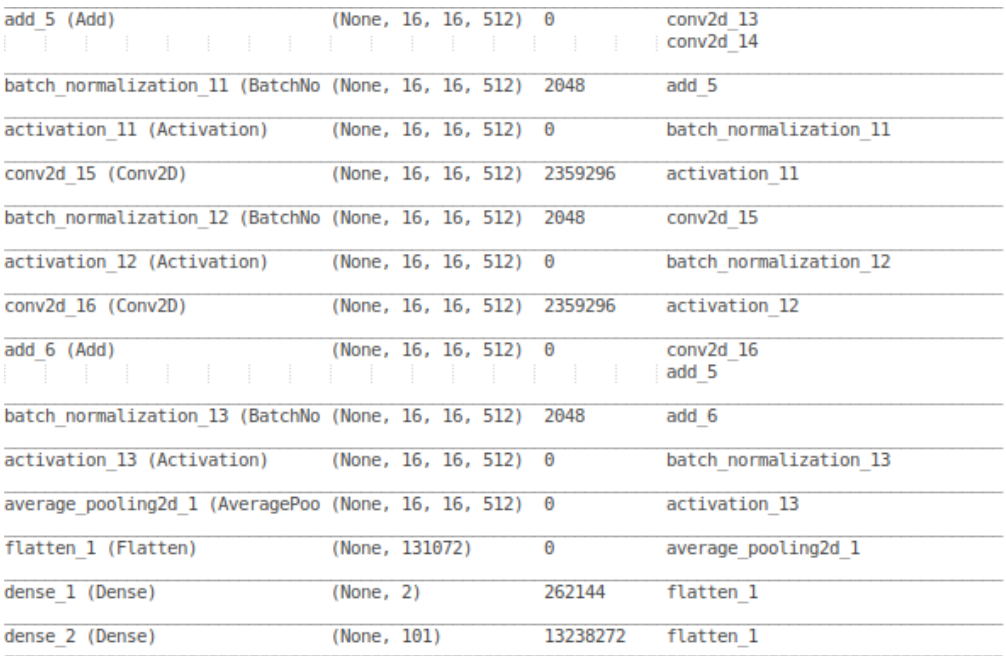
саме на назву попереднього шару від поточного у рядку.



add_3 (Add)	(None, 32, 32, 256)	0	conv2d_8 conv2d_9
batch_normalization_7 (BatchNor	(None, 32, 32, 256)	1024	add_3
activation_7 (Activation)	(None, 32, 32, 256)	0	batch_normalization_7
conv2d_10 (Conv2D)	(None, 32, 32, 256)	589824	activation_7
batch_normalization_8 (BatchNor	(None, 32, 32, 256)	1024	conv2d_10
activation_8 (Activation)	(None, 32, 32, 256)	0	batch_normalization_8
conv2d_11 (Conv2D)	(None, 32, 32, 256)	589824	activation_8
add_4 (Add)	(None, 32, 32, 256)	0	conv2d_11 add_3
batch_normalization_9 (BatchNor	(None, 32, 32, 256)	1024	add_4
activation_9 (Activation)	(None, 32, 32, 256)	0	batch_normalization_9
conv2d_12 (Conv2D)	(None, 16, 16, 512)	1179648	activation_9
batch_normalization_10 (BatchNo	(None, 16, 16, 512)	2048	conv2d_12
activation_10 (Activation)	(None, 16, 16, 512)	0	batch_normalization_10
conv2d_13 (Conv2D)	(None, 16, 16, 512)	2359296	activation_10
conv2d_14 (Conv2D)	(None, 16, 16, 512)	131072	activation_9

Рисунок 4.11 — Архітектура моделі системи

На Рисунку 4.12 зображено два вихідні шари моделі.



add_5 (Add)	(None, 16, 16, 512)	0	conv2d_13 conv2d_14
batch_normalization_11 (BatchNo	(None, 16, 16, 512)	2048	add_5
activation_11 (Activation)	(None, 16, 16, 512)	0	batch_normalization_11
conv2d_15 (Conv2D)	(None, 16, 16, 512)	2359296	activation_11
batch_normalization_12 (BatchNo	(None, 16, 16, 512)	2048	conv2d_15
activation_12 (Activation)	(None, 16, 16, 512)	0	batch_normalization_12
conv2d_16 (Conv2D)	(None, 16, 16, 512)	2359296	activation_12
add_6 (Add)	(None, 16, 16, 512)	0	conv2d_16 add_5
batch_normalization_13 (BatchNo	(None, 16, 16, 512)	2048	add_6
activation_13 (Activation)	(None, 16, 16, 512)	0	batch_normalization_13
average_pooling2d_1 (AveragePoo	(None, 16, 16, 512)	0	activation_13
flatten_1 (Flatten)	(None, 131072)	0	average_pooling2d_1
dense_1 (Dense)	(None, 2)	262144	flatten_1
dense_2 (Dense)	(None, 101)	13238272	flatten_1

Рисунок 4.12 — Архітектура моделі системи

Перший вихідний шар повертає значення віку, другий — статі.

4.1.4. Оцінка якості моделі

Мета машинного навчання у створенні узагальнюючих моделей, які зможуть дати якісний прогноз на даних, що не використовувались у тренуванні моделі [1]. Головною перешкодою для цієї мети є перенавчання . Надзвичайно важливо контролювати це за рахунок надійної оцінки якості узагальнення створюваної моделі. Для оцінювання якості моделі необхідно виконати поділ набору даних на три частини, що називаються — тренувальний набір, перевірочний та контрольний. Під час тренування модель тренується на тренувальному наборі і перевіряється на перевірочному, а після тренування остаточна версія моделі тестується з допомогою контрольного набору. Також при тренуванні моделі відбується налаштування її параметрів, наприклад, вибір шарів, їх типів, послідовності і їх параметрів. Для здійснення цього вибору необхідна постійна перевірка для порівняння. Для цього також використовується перевірочні дані.

Початково дані ділилися тільки на дві частини — тренувальний і тестовий набори. В такому випадку тестовий використовувався для усіх перевірок. При перших спробах створити власну нескладну модель для простої задачі так і пропонується робити. Але виконуючи громістку і серйозну роботу з машинним навчанням, використання одного контрольного (тестового) набору для усіх перевірок: і налаштувань, і кінцевого тестування, призведе до проблем перенавчання, незважаючи на те, що модель не тренується на цих даних. Причиною цього є витік інформації. Кожного разу, коли налаштовується гіперпараметр моделі і оцінка прогнозу виконується на перевірочних даних, допускається певне просочування інформації з цих даних у модель [1]. Одноразове використання не надто шкідливе і залишить перевірочний набор надійним для вимірювання якості моделі, але після декількох виконань цієї операції, усе більша кількість інформації стає відомою моделі. Таким чином, модель неявно навчається на перевірочних даних, що недопустимо при використанні їх для кінцевої оцінки. Тож

тестовий набір повинен складатися з двох: перевірочного, для налаштування архітектури моделі, і контрольного для оцінки якості. Не можна використовувати контрольний набір для яких-небудь налаштувань в моделі, адже тоді оцінка узагальнення моделі точною не буде.

Варто пам’ятати: коли набір даних ділиться на частини, модель втрачає якусь кількість інформації, що могла б використовуватись для її навчання. Тому для тестових наборів не можна брати занадто багато даних, але і малі тестові набори дають менш точну оцінку помилки узагальнення. На практиці використовують пропорції 60:40, 70:30, 80:20 для тренувального та тестового наборів відповідно і 2:1 для перевірочного і контрольного відповідно. Ці значення вибираються з огляду на розмір загального набору. Для надто великих датасетів навіть умісно використовувати пропорції 90:10 та 99:1.

Відкидати тестові дані після завершення створення моделі не рекомендується — варто перетренувати останній раз готову модель на усьому наборі, щоб вона отримала максимум доступної інформації.

4.2. Структура бази даних

База даних системи має невелику кількість сутностей, нескладну структуру, але складність полягає в тому, що система регулярно опрацьовує повний вміст даних в базі, проводить складні розрахунки і операції з ними. Проектована база даних пройшла процедуру нормалізації і відповідає усім нормальним формам реляційних баз даних:

— перша нормальна форма: всі атрибути відношення містять тільки атомарні значення;

— друга нормальна форма: відношення зведено до першої нормальної форми, та первинний ключ однозначно визначає кортеж;

— третя нормальна форма: відношення зведено до другої нормальної форми, і неключові атрибути не мають транзитивних залежностей.

Усього база даних має три відношення — три таблиці, дві з яких є словарями значень віку і статі. Третя таблиця містить основну частину даних — інформація про користувачів додатка. Словарі — таблиці ages і genders, основна таблиця — users (рисунок 4.13).

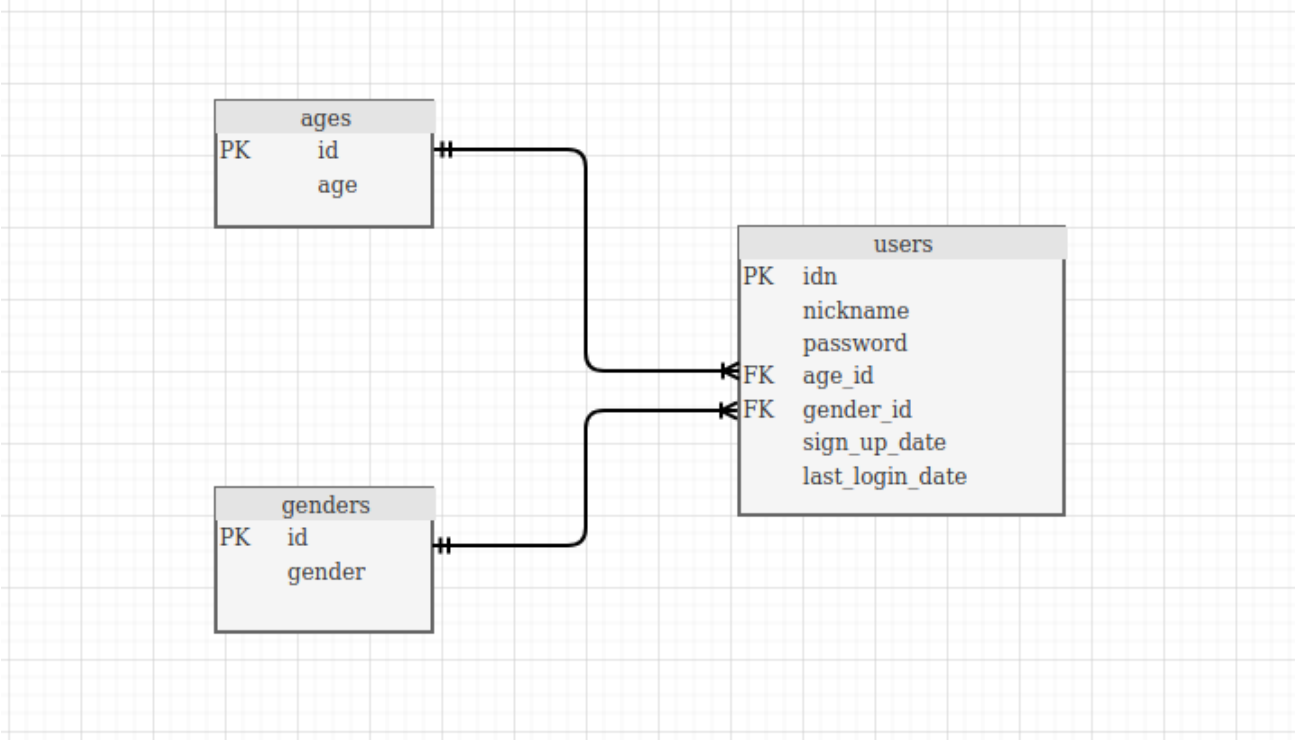


Рисунок 4.13 — Структура бази даних

Кожна таблиця має стовпчик ідентифікаційний коду кожного запису, що є primary key (первинним ключем) для таблиці. В таблиця-словарях ages і genders він має назву id, а в таблиці users — idn, в усіх таблицях тип атрибута Integer і він визначений як автоінкрементний. Структура словарів проста — вони мають серед своїх стовпців лише ідентифікаційний код і назву групи — вікової (наприклад, від 14 до 23, або від 48 до 59) і статевої (наприклад, Female, Male). Назви груп у словарях називаються відповідно до назв таблиць — age і gender для ages і genders відповідно, і мають тип String(10), нульове значення неможливе. Таблиця users окрім вже описаного атрибута idn має nickname, що визначає унікальне ім’я користувача в системи, має тип

String(30) і нульове значення для нього неможливе. Наступний атрибут — password з типом String(32), що містить закодований пароль користувача у вигляді 32 символів. Атрибути age_id та gender_id є зовнішніми ключами для зв’язку багато-до-одного з таблицями словарями ages і genders відповідно. Поля можуть мати нульове значення, оскільки не про всіх користувачів відома інформація про вік і стать, на заповнення цих даних необхідний деякий час користування системою. Поля sign_up_date та last_login_date мають тип DateTime, нульове значення неможливе, бо мають значення по замовчуванню — поточну дату і час. При реєстрації користувача у поля sign_up_date та last_login_date вносяться дати реєстрації та останнього входу в систему. Для першого використання додатка ці дати співпадають, надалі при кожній реєстрації дата останнього входу оновлюється.

4.3. Реалізація інтерфейсу

Інтерфейс користувача системи був реалізаваний з допомогою бібліотеки створення графічного інтерфейсу Kivy. Було використано багато методів і вбудованих класів бібліотеки для створення сутностей — елементів інтерфейсу, таких як екран додатку, вікно, плаваюче вікно, випливаючий список, поле вводу, кнопка, зображення, текстура для відеопотоку з камери тощо. Також було використано додатковий механізм бібліоткеки — мова Kvlanguage. З її допомогою оформлення і дизайн усіх сутностей розширюють свої можливості, а також є можливість відокремити представлення у файли з розширенням .kv від логіки інтерфейсу у python-файлах. Кожен екран додатка мітситься в окремому файлі з відповідною назвою, наприклад:

— початковий екран — файли home_screen.py і home_screen.kv;

— екран для функціонала користувача — user_screen.py і user_screen.kv;

— екран для функціонала адміністратора системи — admin_screen.py і

admin_screen.kv.

Деякі елементи мають більш широке представлення і надто громістку реалізацію класів і їх методів, тож їх віднесено в окремі файли. Наприклад, допоміжні елементи екрану користувача винесені у файл user_screen_properties.kv.

Система має різний функціонал для двох типів користувачів — для звичайного користувача і адміністратора системи. Функції, можливі для виконання користувачем системи, описані на use-case діаграмі (рисунок 4.14).

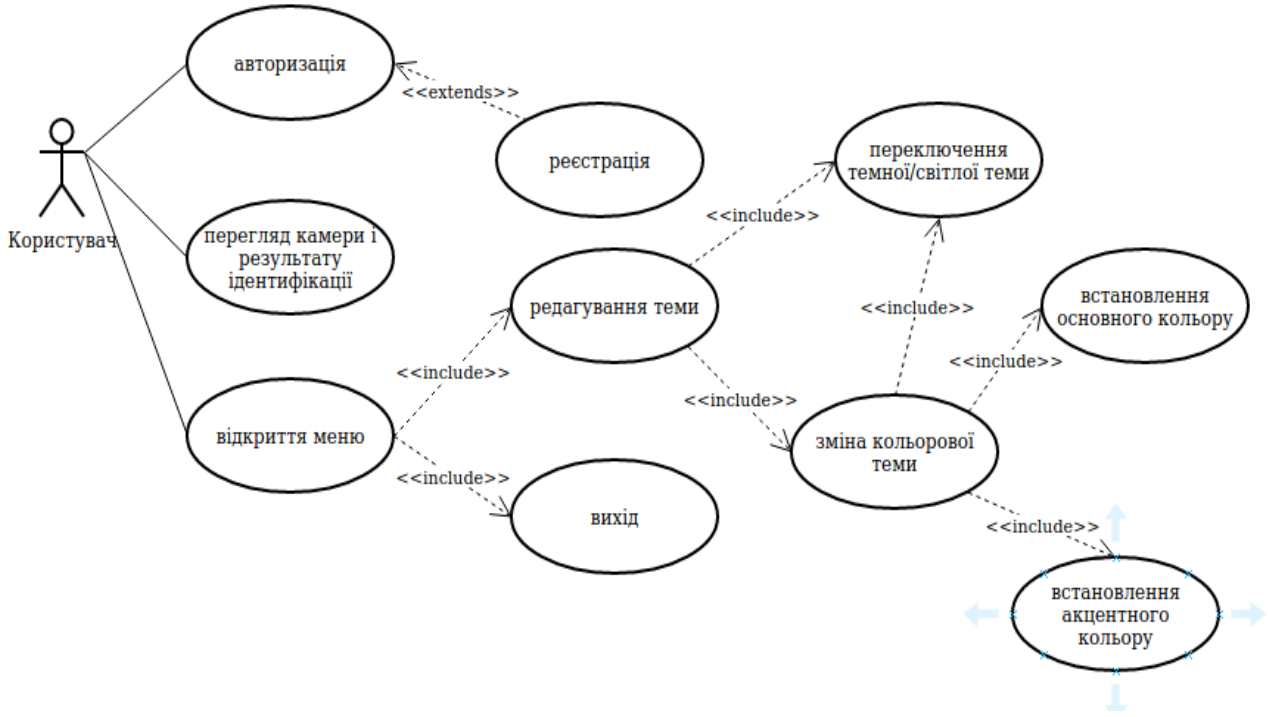


Рисунок 4.14 — Use-case діаграма для користувача системи

Особливістю екрана користувача перед екраном адміністратора є меню зміни теми інтерфейсу.

Функціонал адміністратора зображений на діаграмі прецедентів (рисунок 4.3). Цей актор системи працює з даними. Таблиця, яку він може переглядати є таблицею , описаною в попередньому підрозділі. З бази даних дублюються усі записи таблиці з виконанням команди JOIN, замінюючи таким чином ідентифікатор-ключ id до таблиці ages на назву вікової групи

age і аналогічно з таблицею genders.

Рисунок 4.15 — Use-case діаграма для адміністратора системи

Ні одна операція редагування таблиці не виконується, поки усі обов’язкові поля не будуть заповнені адміністратором і заповнені коректно відповідно до формату даних у базі.

4.4. Тестування системи

Для тестування системи ідентифікації людей за віком і статтю використовувалися unit-тести або автономне (модульне) тестування на мові програмування Python.

Для перевірки налагодженої і коректної роботи програмного забезпечення проводилося тестування основних модулей програми: моделі машинного навчання, класу роботи з базою даних, користувацького інтерфейсу системи.

На Рисунку 4.16 приводиться результати виконання коду тестувального модуля системи, запущеного у програмі PyCharm Professional. Вивід роботи

програми демонструє успішні результати тестування системи.

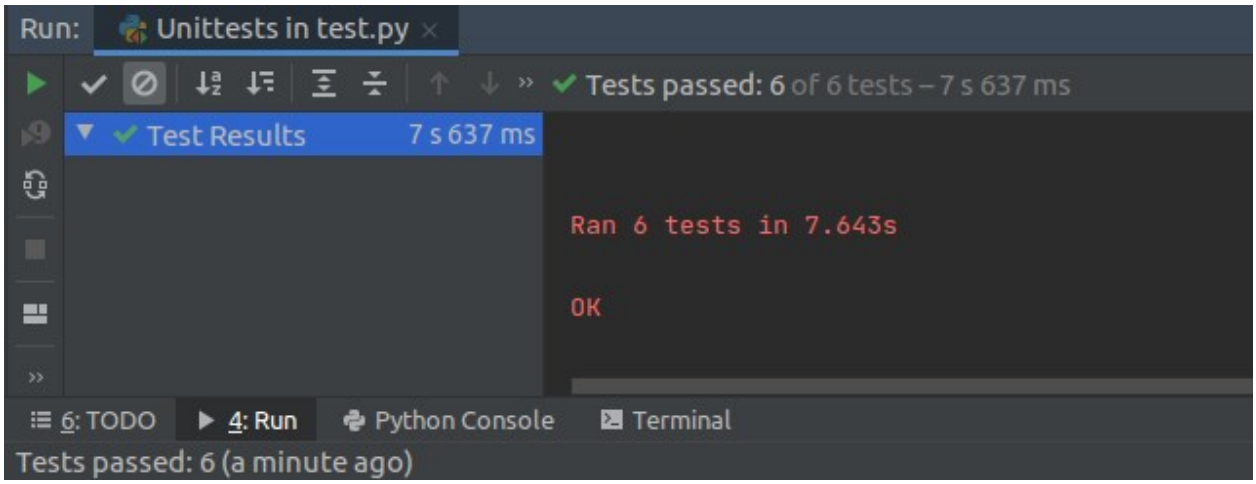


Рисунок 4.16 — Результати тестування системи

4.5. Висновки до розділу

В розділі представлено опис реалізації програмного продукту, який використовує глибоке навчання для ідентифікації людей на відео за віком і статтю. Описано основні етапи розробки системи, такі як створення моделі машинного навчання, реалізацію інтерфейсу і тестування системи. У вигляді схеми описано етапи створення моделі машинного навчання. Представлено архітектуру широкої залишкової нейронної мережі з детальним описом усіх складових і операції мережі, обґрунтовано усі переваги даної архітектури. Проведено детальний аналіз етапу підготовки набору даних, описаний датасет IMDB-Wiki, що був обраний для тренування моделі, та проаналізовано способи оцінки якості моделей машинного навчання і їх значення у процесі створення системи. Описано структуру створеної бази даних, представлено діаграми прецедентів для опису функціонала додатку для усіх існуючих акторів системи та результати тестування програмного забезпечення.

5. ОПИС РОБОТИ КОРИСТУВАЧА З СИСТЕМОЮ

5.1. Робота користувачів з інтерфейсом системи

Обидва актора системи: як звичайний користувач, так і адміністратор, повинні дотримуватись усіх вимог роботи з користувацьким інтерфейсом додатку для коректної роботи усіх модулів системи. Додаток має початковий екран, з якого можна перейти до авторизації в системі (рисунок 5.1). У верхній частині вікна зображен логотип та назва системи.

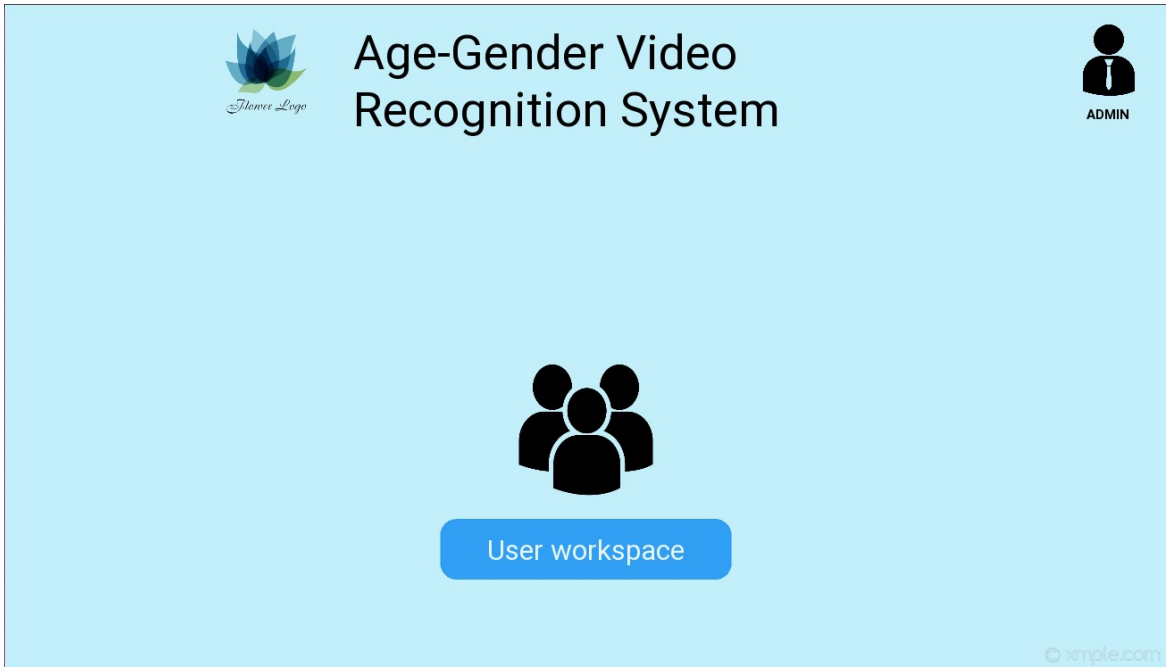


Рисунок 5.1 — Початковий екран

Екран має дві кнопки — вхід для адміністратора системи і окремо вхід для користувача, що мають інтуїтивно зрозумілий дизайн і розміщення. Після нажаття на кнопку з підписом “Admin” відкривається вікно авторизації (рисунки 5.2-5.3). Вікно має два поля для вводу імені користувача і пароля та

кнопку “login” для переходу до аутентифікації. Поле для введення паролю має властивість закриття символів зірочками (рисунок 5.3).

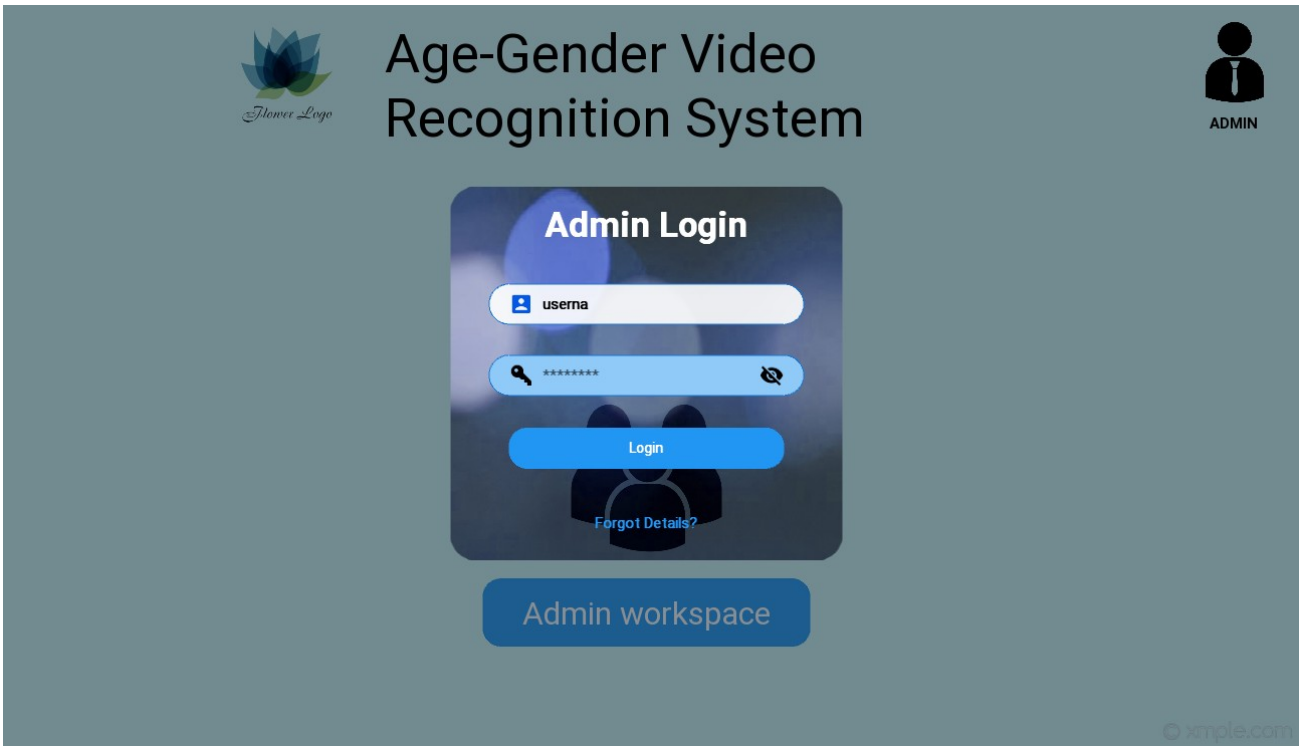


Рисунок 5.2 — Авторизація адміністратора: заповнення поля username

Після вдалої авторизації адміністратор потрапляє на головний екран середовиа роботи з даними. Відразу на першому екрані розміщена таблиця користувачів (рисунок 5.4). Зліва знаходиться меню з кнопкою виходу на початковий екран. В правому верхньому кутку розташоваша кнопка “плюс” для додавання запису в таблицю користувачів. На наступному Рисунку 5.5 зображена панель для додавання запису. Панель має наступні поля для введення: nickname, password, sign up date, last login date, gender, age. Оскільки поля gender і age мають містити групу за віком і статтю, які визначені у окремих таблицях бази даних, тож кількість варіантів введення і формат обмежені. Щоб користувач не помилився і вставка у таблицю відбулася вдало, при нажатті на одне з цих полів з’являється вікно для вибору групи. А ввести в це поле вільний текст, що не відповідає

встановленому формату, неможливо (рисунок 5.6). Сама ж таблиця містить записи з інформацією про користувачів.

Рисунок 5.3 — Авторизація адміністратора: заповнення поля пароля

Зліва запис має кнопки редагування та видалення(рисунки 5.4 — 5.9).

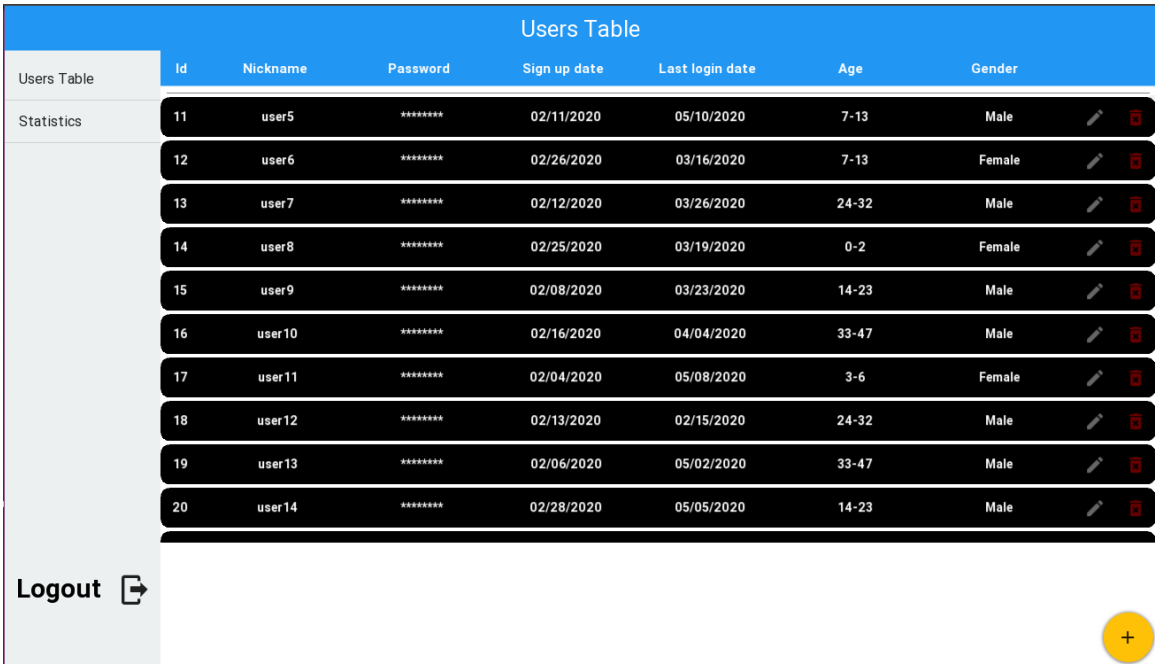


Рисунок 5.4 — Екран адміністратора: перегляд таблиці користувачів

При редагування запису таблиці (рисунок 5.7) знизу панель, як і при з

початково заповненими поточними значеннями полів.

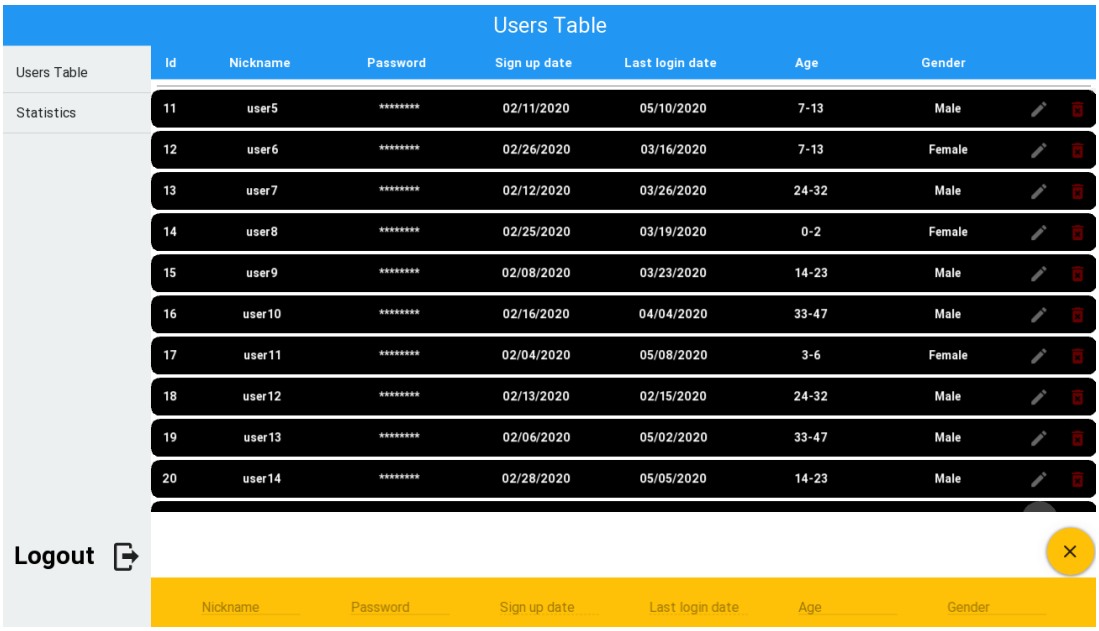


Рисунок 5.5 - Додання нового запису до таблиці

Після закінчення введення даних необхідно нажати кнопку пташки.

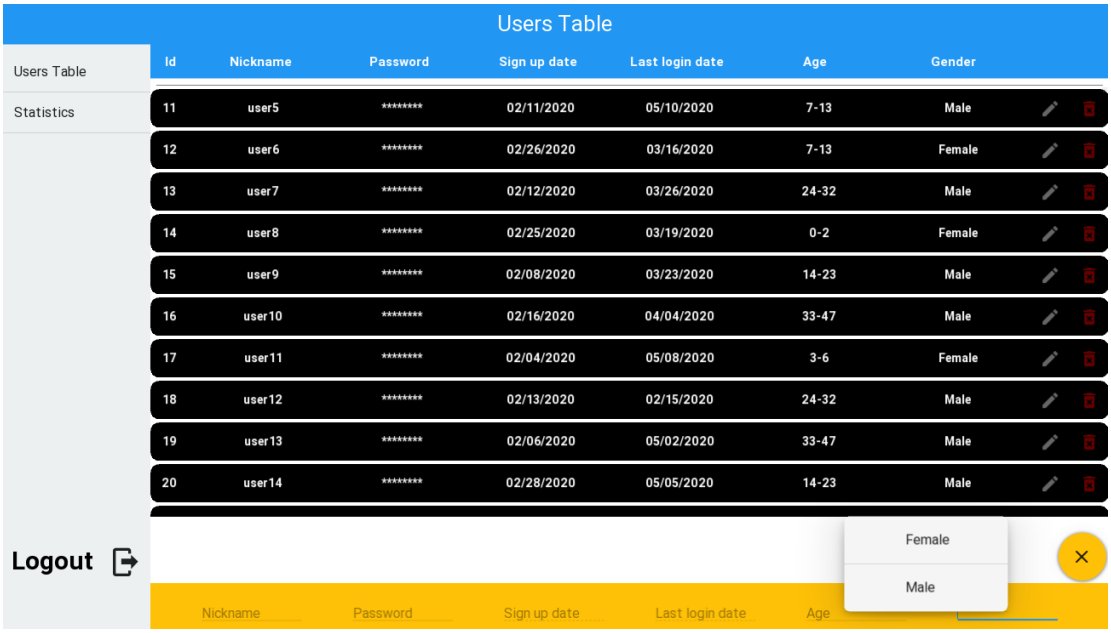


Рисунок 5.6 — Заповнення полів вікової групи і статі

Робити які-небудь зміни в таблиці під час її редагування неможливо. Наприклад, намагаючись видалити рядок, поки таблиця редагується і нижня

панель відкрита, адміністратор отримає повідомлення про це (рисунок 5.8).

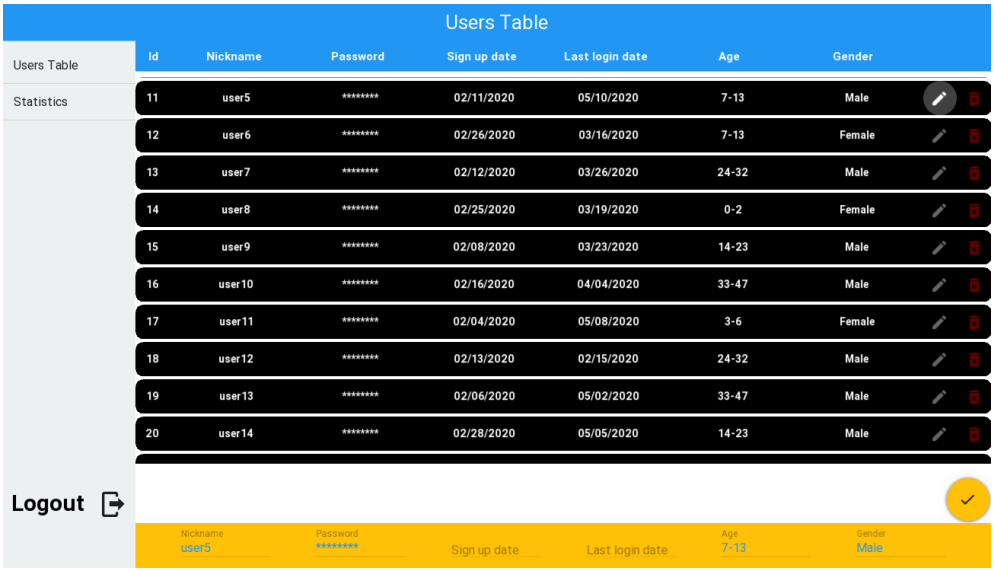


Рисунок 5.7 — Редагування запису в таблиці

За умови, що таблиця не редагується, можна безпечно видалити рядок.

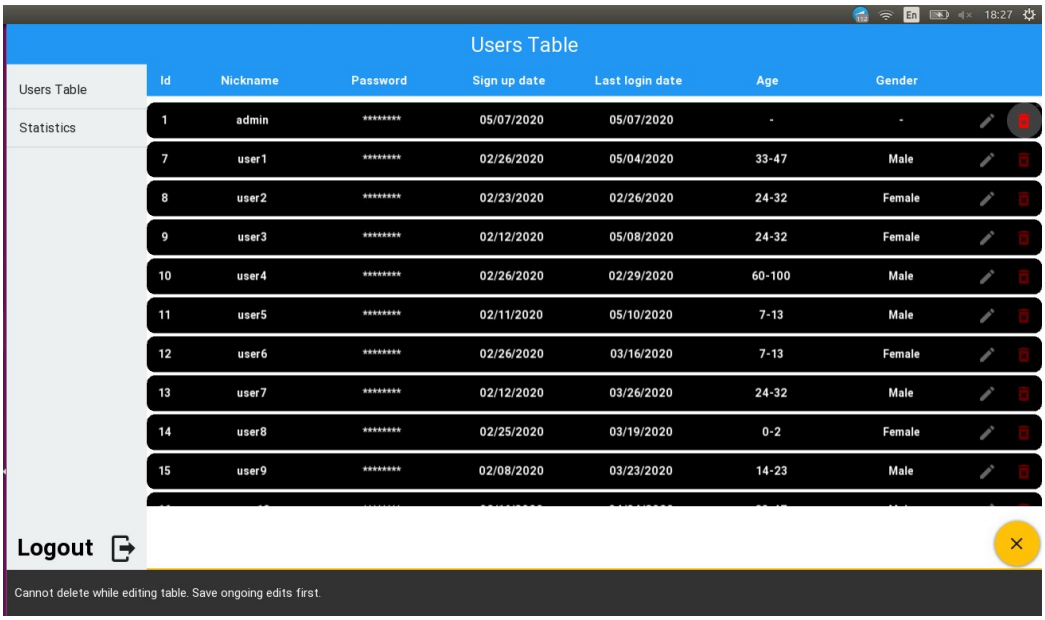


Рисунок 5.8 — Видалення рядка під час редагування таблиці

Для видалення необхідно нажати на червону кнопку права на записі. Після цього система відкриє окреме вікно для підтвердження видалення

(рисунок 5.9), вказавши ім’я користувача, якого необхідно видалити.

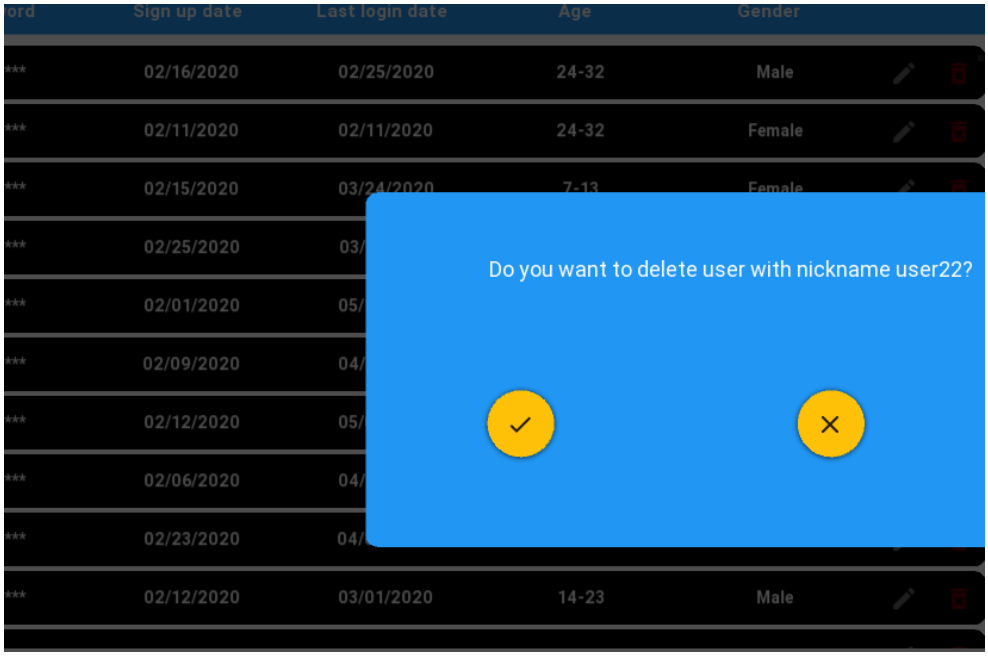


Рисунок 5.9 — Підтвердження видалення запису

Якщо видалення пройшло успішно, система повідомить (рисунок 5.10).

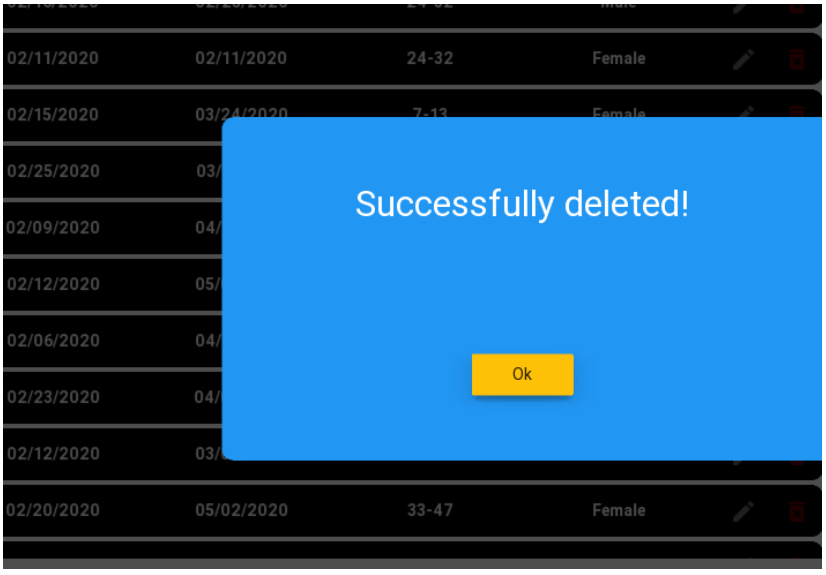


Рисунок 5.10 — Успішне видалення

Перед тим як продовжити будь-яку роботу на екрані, завжди необхідно завершити введення даних у нижню панель. Під час редагування таблиці

неможливим також є перехід у інше вікно (рисунок 5.11).

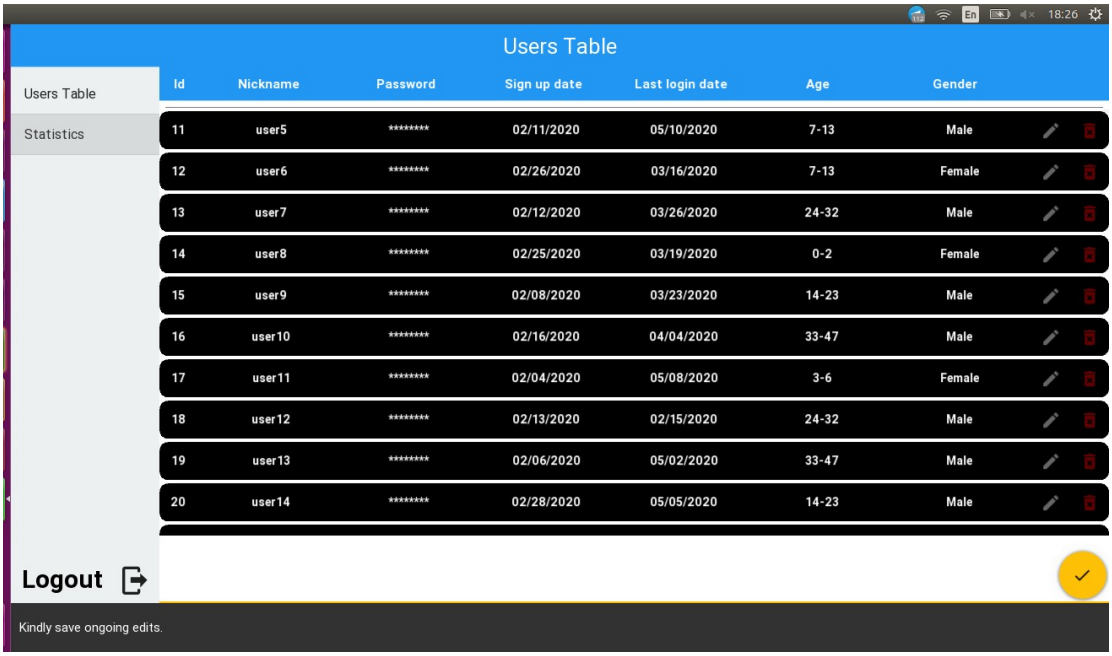


Рисунок 5.11 — Перехід у інше вікно під час редагування

Сторінка адміністратора має ще одне вікно — перегляд статистики щодо розподілу користувачів (Рисунок 5.12).

Рисунок 5.12 — Екран перегляду статистики

На цьому вікні зображені деякі графіки щодо розподілу віку, статі

користувачів, також інформація щодо частоти їх реєстрації та використання додатку. Вхід користувача в систему відбувається як на Рисунку 5.13.

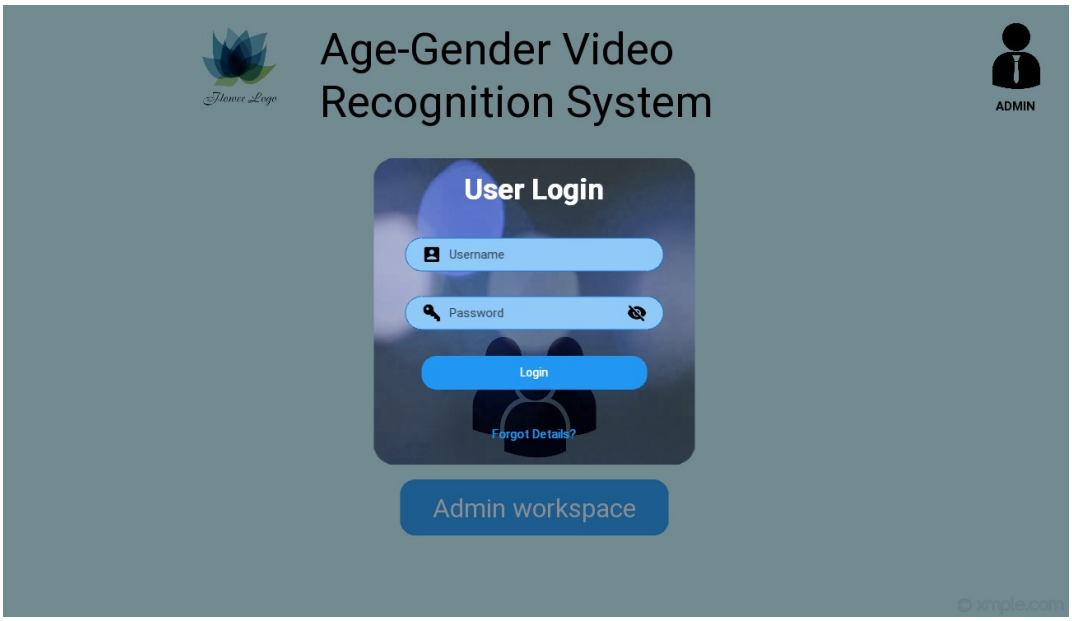


Рисунок 5.13 — Авторизація користувача

Головний екран для користувача — це відкрите зображення з веб-камери, де має бути зображене обличчя користувача (рисунок 5.14).

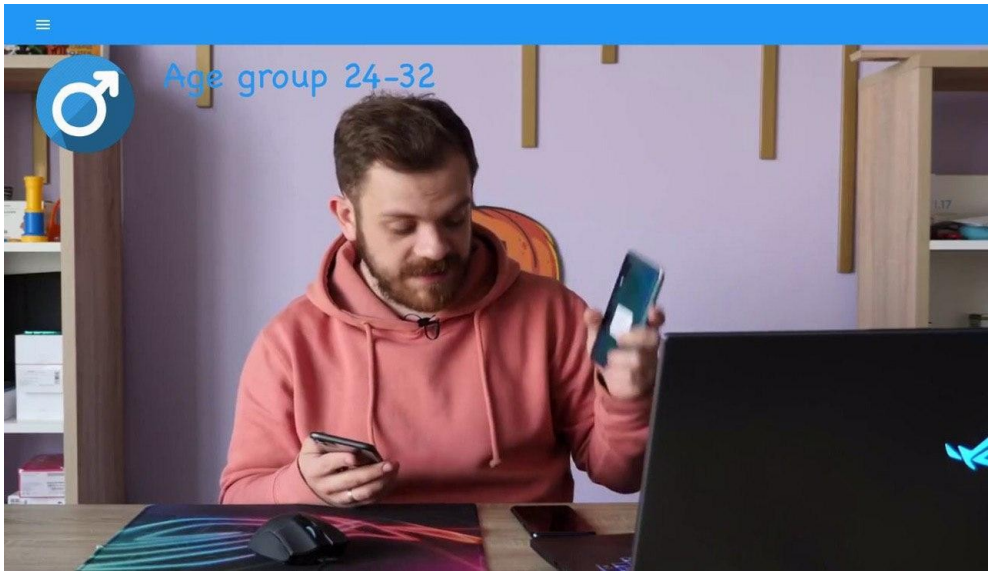


Рисунок 5.14 — Екран ідентифікації користувача за віком і статтю

На екрані з’явиться поточна оцінка віку і статі людини, зображеної на відео.

Екран користувача також має меню(рисунок 5.15).

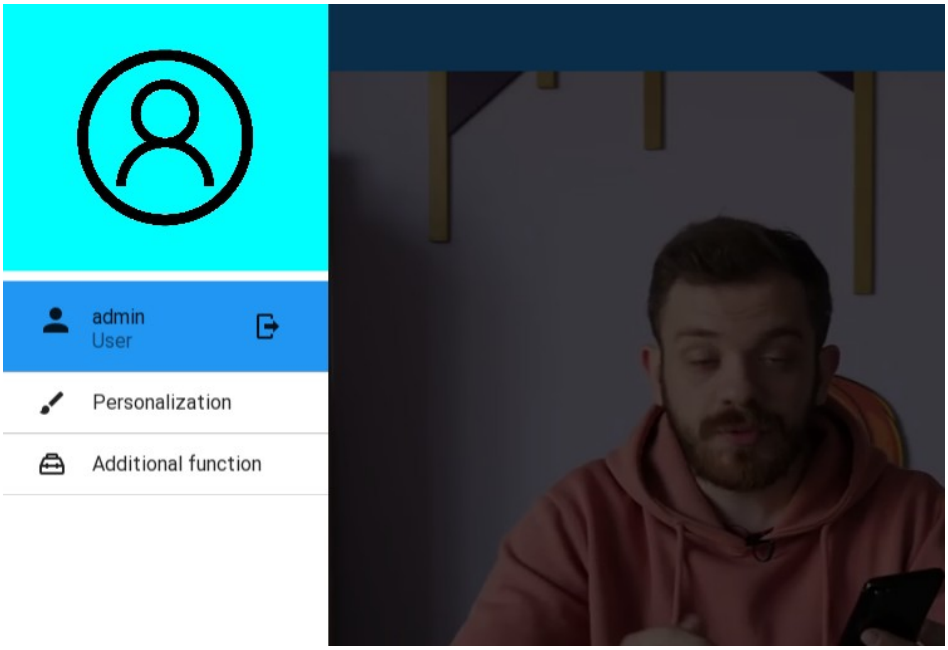


Рисунок 5.15 — Меню на екрані користувача

Користувач з цього меню має можливість вийти з системи, змінити кольорову тему (рисунки 5.16).

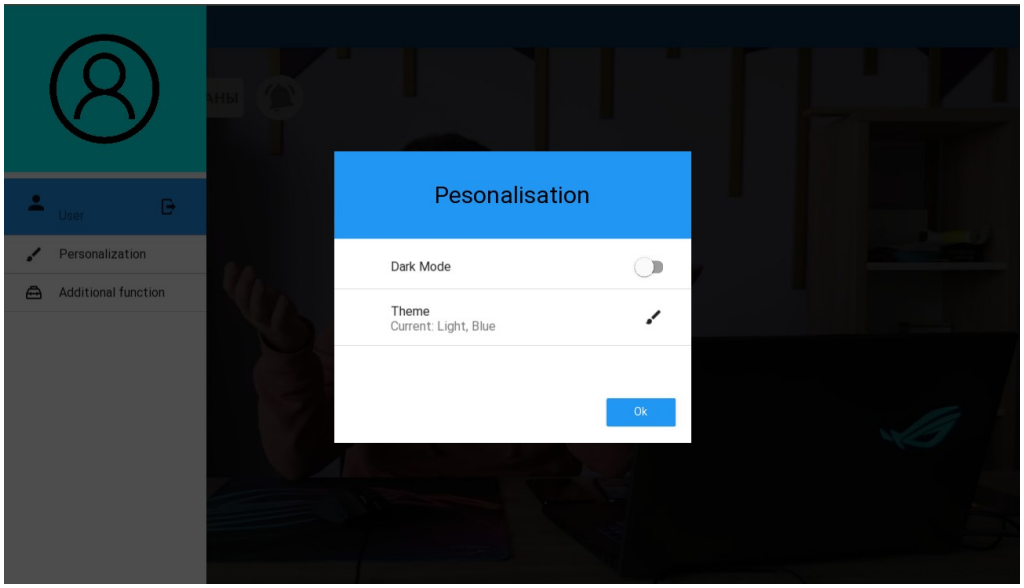


Рисунок 5.16 — Вікно форматування теми інтерфейсу

Для цього йому представлений достатній вибір кольорів у меню (рисунок

5.17). Також користувач може обрати темне або світле оформлення інтерфейсу (рисунок 5.19).

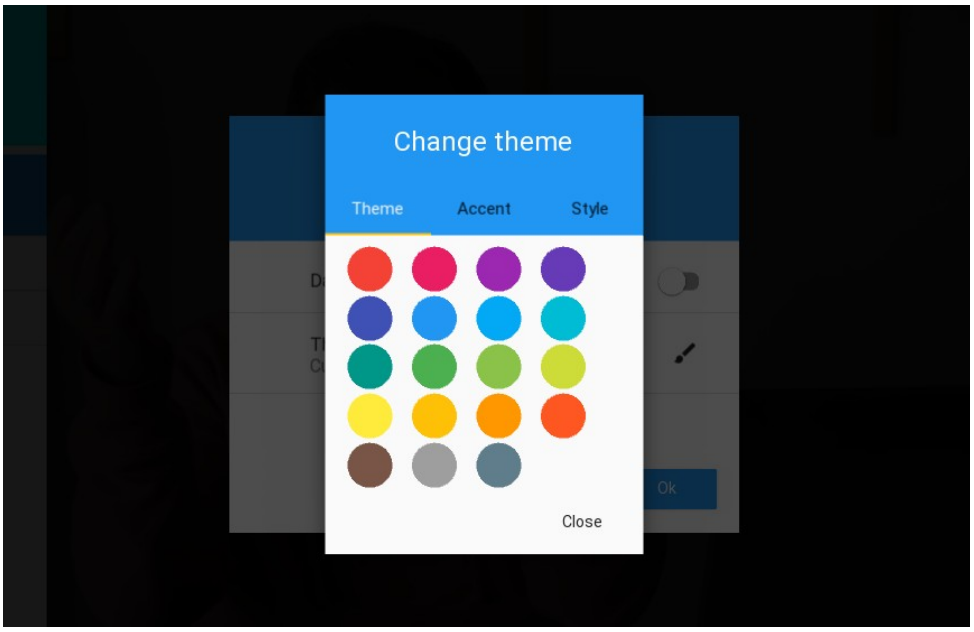


Рисунок 5.17 — Вибір кольорової теми

Результат зміни теми інтерфейсу представлений на рисунку 5.18.

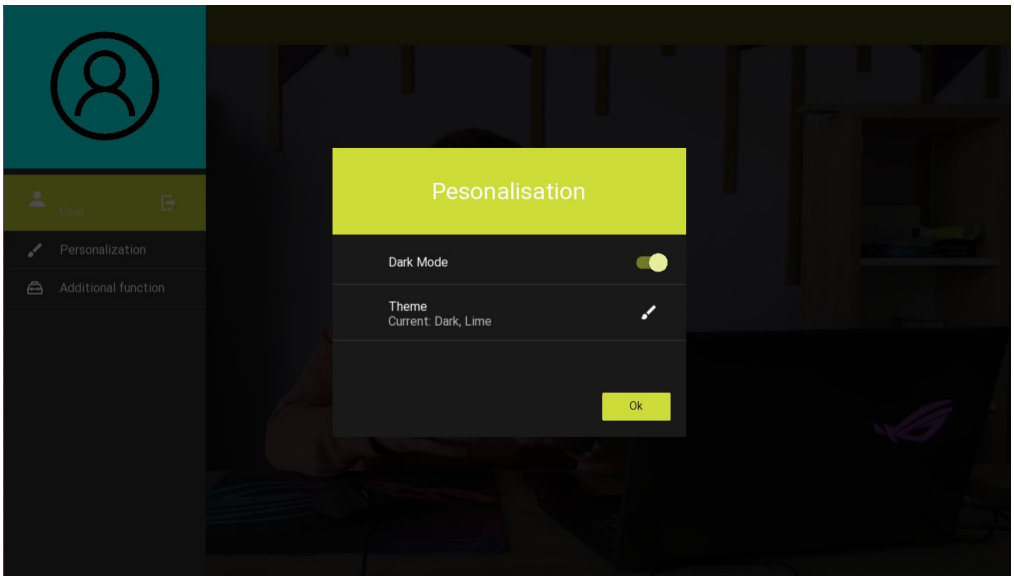


Рисунок 5.18 — Зміна теми інтерфейсу зі світлої на темну

Отже, інтерфейс користувача гарантує широкі можливості і стильний дизайн.

5.2. Запуск програмного забезпечення

Збірка проекту була реалізована з допомогою докер-контейнера, що гарантує успішний запуск програми на будь-якій операційній системі.

Приклад успішного запуску (з інтерфесу командної строки на операційній системі Linux Ubuntu) представлено на Рисунку 5.19.

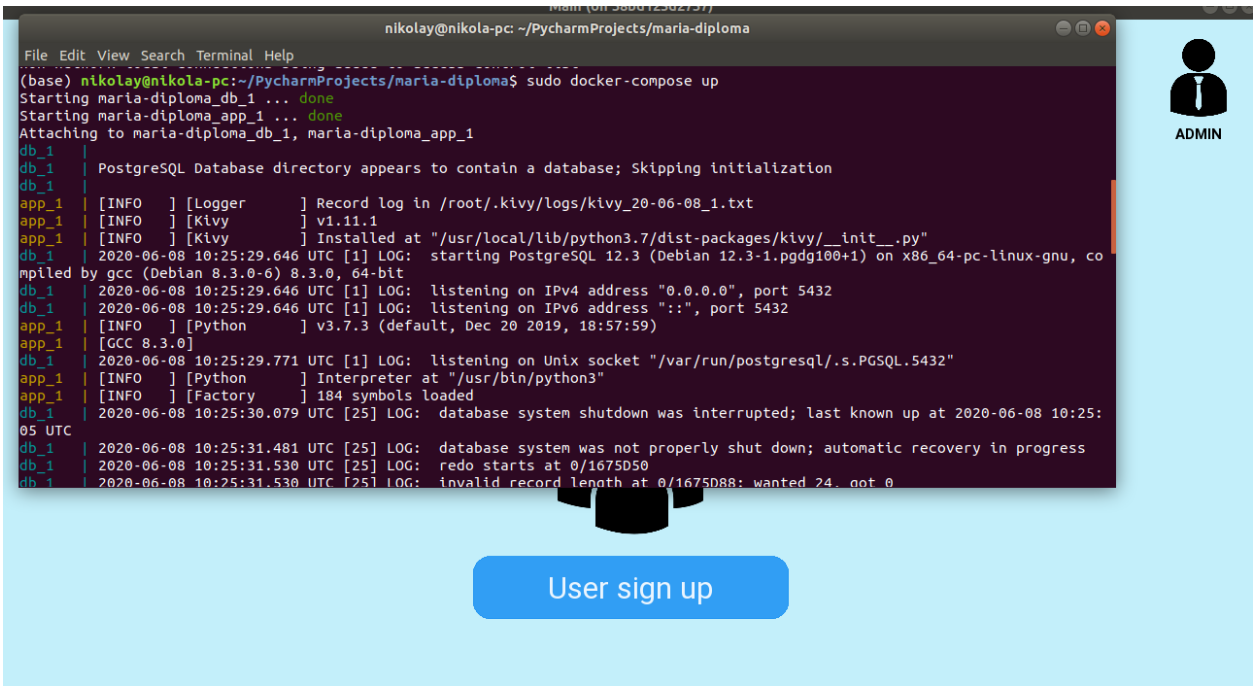


Рисунок 5.19 — Успішний запуск системи з docker-контейнера

Єдиними вимогами для безперебійного запуску і роботи програмного забезпечення є наявність підключеної веб-камери і налаштування програми Docker. Запуск програмного забезпечення відбувається за допомогою команди “docker-compose up”. При цьому контейнер запускає два мікросервіса системи: першим базу даних, другим сам додаток.

5.3. Висновки до розділу

В розділі було представлено усі необхідні користувачу вимоги для зауску і використання розробленого в ході виконання дипломної роботи програмного забезпечення. Було продемонстровано приклад успішного запуску системи з використанням docker-контейнера.

Система має два актори — користувач і адміністратор, тож роботу з графічним інтерфейсом додатку було описано окремо для кожного актора, також представлено усі можливості, які система пропонує користувачеві — це зручна і надійна авторизація в системі, перегляд даних у вигляді таблиці, їх редагування, перегляд графіків, робота з камерою, налагодження теми інтерфейсу.

ВИСНОВКИ

Розроблене програмне забезпечення за темою дипломної роботи “Аналіз відеопотоку: ідентифікація людей за статтю та віковою групою” має за мету оброляти відеоматеріали і визначати вікову групу та стать людини, що зображена на них. Для досягнення мети було виконано ряд етапів, на які була поділена основна задача:

1. пошук візуальних даних для вивчення і аналізу природи даних з метою класифікації зображень;
2. розробка моделі нейронної мережі, налагодження навчання і роботи моделі;
3. проектування бази даних системи;
4. розробка алгоритму роботи сисмети;
5. написання програмного коду для створення користувацього інтерфейсу;
6. тестування і налагодження роботи програмного продукту;
7. створення докер-контейнера і розгортка додатку.

Було проаналізовано різні архітектури моделей машинного навчання для знаходження оптимальної для виконання поставленої задачі, були проведені дослідження даних перед поданням їх на вивчення моделлю, проведена необхідна обробка.

В результаті виконання дипломної роботи було розроблено програмний продукт для розпізнавання осіб, який здатен виявляти морфологічні критерії людей, такі як вік і стать, шляхом аналізу відеопотоку в режимі реального часу і обробляти отримані дані шляхом аналізу розподілу, надаючи усі результати користувачу. Завдяки такій інноваційній розробці, додатки матимуть можливість автоматизовано збирати дані про користувачів, а розробники налаштовувати контент в залежності від розподілу, таким чином завжди орієнтуючись на своїх клієнтів.

Список використаних джерел

1. Шолле Ф. Глубокое обучение на Python / Франсуа Шолле. – СПб.: Питер, 2018. – 400 с. – (Библиотека программиста).

2. Рашка С. Python и машинное обучение / Себастьян Рашка. – Москва: ДМК Пресс, 2017. – 418 с.

3. Ibrahim Mousa Al-Zuabi. Predicting customer’s gender and age depending on mobile phone data [Електронний ресурс] / Ibrahim Mousa Al-Zuabi, Assef Jafar, Kadan Aljoumaa. – 2019. – Режим доступу до ресурсу: https://journalofbigdata.springeropen.com/articles/10.1186/s40537-019-0180-9.

4. Mark Lutz. Learning Python / Mark Lutz. – Sebastopol, United States: O'Reilly Media, 2013.

5. Robert Laganiere. OpenCV 3 Computer Vision Application Programming Cookbook / Robert Laganiere. – Birmingham: Packt Publishing, 2017. – 444 с.

6. Paul DuBois. MySQL Cookbook / Paul DuBois., 2002. – 1006 с.

7. Roberto Ulloa. Kivy: Interactive Applications in Python / Roberto Ulloa. – Birmingham: Packt Publishing, 2013. – 122 с.

8. Рой О. Искусство автономного тестирования с примерами на C# / Ошероув Рой. – Москва: ДМК Пресс, 2014. – 360 с.

9. Моуэт Э. Использование Docker / Эдриен Моуэт. – Москва: ДМК Пресс, 2017. – 354 с.

10. Rasmus Rothe. Deep expectation of real and apparent age from a single image without facial landmarks [Електронний ресурс] / Rasmus Rothe, Radu Timofte, Luc Van Gool. – 2016. – Режим доступу до ресурсу: https://data.vision.ee.ethz.ch/cvl/publications/papers/articles/eth_biwi_01299.pdf.

11. Rodolfo B. Machine Learning for Developers / Bonnin Rodolfo. – Birmingham: Packt Publishing, 2017. – 270 с.

12. Deep Residual Learning for Image Recognition [Електронний ресурс] / Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. – 2019. – Режим доступу до ресурсу:https://www.cv-foundation.org/openaccess/content_cvpr_2016/papers/He_Deep_Residual_Learning_CVPR_2016_paper.pdf.

13. Sergey Zagoruyko. Wide Residual Networks [Електронний ресурс] / Sergey Zagoruyko, Nikos Komodakis. – 2017. – Режим доступу до ресурсу: https://arxiv.org/pdf/1605.07146.pdf.

ДОДАТОК 1

Аналіз відеопотоку: ідентифікація людей за статтю та віковою групою

Специфікація

УКР.НТУУ “КПІ ім. Ігоря Сікорського”_ТЕФ_АПЕПС_ТВ6127_20Б

Аркушів 2

Київ - 2020

Позначення	Найменування	Примітки
Документація		
УКР.НТУУ “КПІ ім. Ігоря Сікорського”_ТЕФ_АПЕПС_ТВ6127_20Б 81-1	Герасимова М.В. Записка.doc	Пояснювальна записка
Компоненти		
УКР.НТУУ “КПІ ім. Ігоря Сікорського”_ТЕФ_АПЕПС_ТВ6127_20Б 12-1	model/wide_resnet.py	Модуль з класом моделі машинного навчання
УКР.НТУУ “КПІ ім. Ігоря Сікорського”_ТЕФ_АПЕПС_ТВ6127_20Б 12-2	model/ trainde_model.py	Модуль для ідентифікації людини на зображенні
УКР.НТУУ “КПІ ім. Ігоря Сікорського”_ТЕФ_АПЕПС_ТВ6127_20Б 12-3	db.py	Модуль для роботи з базою даних системи: створення таблиць і роботи з ними
УКР.НТУУ “КПІ ім. Ігоря Сікорського”_ТЕФ_АПЕПС_ТВ6127_20Б 12-4	gui/user_screen.py gui/admin_screen.py gui/popups.py gui/home_screen.py	Модулі для реалізації графічного інтерфейсу користувача

ДОДАТОК 2

Аналіз відеопотоку: ідентифікація людей за статтю та віковою групою

Текст програмного модуля

УКР.НТУУ “КПІ ім. Ігоря Сікорського”_ТЕФ_АПЕПС_ТВ6127_20Б 12-1

Аркушів 7

Київ - 2020

```
import logging
import sys
import numpy as np
from keras.models import Model
from keras.layers import Input, Activation, add, Dense, Flatten, Dropout
from keras.layers.convolutional import Conv2D, AveragePooling2D
from keras.layers.normalization import BatchNormalization
from keras.regularizers import l2
from keras import backend as K

sys.setrecursionlimit(2 ** 20)
np.random.seed(2 ** 10)


# клас моделі - широкої залишкової мережі
class WideResidualNetwork:
    # метод ініціалізації атрибутів класу
    # при створенні нового екземпляру класу
    # приймає аргументи:
    # - розмірність зображень,
    # які подаватимуться на вхід до мережі
    # - глибина мережі
    # - параметр k для широкого залишкового блока
    def __init__(self, image_size, depth=16, k=8):
        self._depth = depth
        self._k = k
        self._weight_decay = 0.0005
        self._use_bias = False
        self._weight_init = "he_normal"
```

```
        # встановлення розмірності для вхідних даних
        # в залежності від фреймворка (theano або tensorflow),
        # який використовує бібліотека keras
        if K.common.image_dim_ordering() == "th":
            # фреймворк theano
            logging.debug("image_dim_ordering = 'th'")
            self._channel_axis = 1
            self._input_shape = (3, image_size, image_size)
        else:
            # фреймворк tensorflow
            logging.debug("image_dim_ordering = 'tf'")
            self._channel_axis = -1
            self._input_shape = (image_size, image_size, 3)

    # метод створення базового або 'тонкого з'єднання' залишкового блока з
шарів мережі
    def _wide_block(self, n_input_plane, n_output_plane, stride):
        # результуюча функція, яку поверне метод
        def f(net):
            # параметри для створення блока
            # kernel: (ширина, довжина) ядра згорткового шара
            # stride: розміри кроку ядра (по вертикалі, по горизонталі)
            # pading: тип відступу
            conv0_layer_params = {'kernel': (3,3), 'stride': stride, "padding": "same"}
            conv1_layer_params = {'kernel': (3,3), 'stride': (1,1), "padding": "same"}

            n_bottleneck_plane = n_output_plane

            # в залежності від розмірності вхідних і вихідних даних,
```

```
        # визначається чи використати згортковий шар з ядром (1, 1)
        if n_input_plane != n_output_plane:
            use_shortcut_conv_layer = True
        else:
            use_shortcut_conv_layer = False

        # створення залишкового блока
        # перший згортковий шар
        block = BatchNormalization(axis=self._channel_axis)(net)
        block = Activation("relu")(block)

        # у випадку додавання згорткового шару з ядром  (1,1)
        # у змінну net запам'ятовується поточний шар
        if use_shortcut_conv_layer:
            net = block

        block = Conv2D(n_bottleneck_plane,
kernel_size=conv0_layer_params['kernel'],
                       strides=conv0_layer_params['stride'],
                       padding=conv0_layer_params['padding'],
                       kernel_initializer=self._weight_init,
                       kernel_regularizer=l2(self._weight_decay),
                       use_bias=self._use_bias)(block)

        # другий згортковий шар
        block = BatchNormalization(axis=self._channel_axis)(block)
        block = Activation("relu")(block)
        # block = Dropout(self._dropout_probability)(block)
        block = Conv2D(n_bottleneck_plane,
```

```
kernel_size=conv1_layer_params['kernel'],
                        strides=conv1_layer_params['stride'],
                        padding=conv1_layer_params['padding'],
                        kernel_initializer=self._weight_init,
                        kernel_regularizer=l2(self._weight_decay),
                        use_bias=self._use_bias)(block)

        # чергуючи блоки, використовується згортковий шар (1,1)
        # для реалізації тонкого з'єднання
        if use_shortcut_conv_layer:
            # згортковий шар поєднується з проміжним, що був запам'ятований
раніше
            shortcut = Conv2D(n_output_plane, kernel_size=(1, 1),
                              strides=stride,
                              padding="same",
                              kernel_initializer=self._weight_init,
                              kernel_regularizer=l2(self._weight_decay),
                              use_bias=self._use_bias)(net)
        else:
            # в іншому випадку відбувається з'єднання з вхідним блоком шара
            shortcut = net
        # зв'язок відбувається з використанням функції суми
        return add([block, shortcut])

    return f

def _layer(self, block, n_input_plane, n_output_plane, stride):
    # метод додає 2 залишкових блока
    # в другому для тонкого з'єднання використовується згортковий шар з
```

```
ядром (1, 1)
        def f(net):
            net = block(n_input_plane, n_output_plane, stride)(net)
            net = block(n_output_plane, n_output_plane, stride=(1, 1))(net)
            return net
        return f


    def __call__(self):
        logging.debug("Creating model...")
        # мережа складається з трьох груп по два залишкових блока
        # перевірка відповідності заданої глибини мережі і кількості блоків у
групі
        assert ((self._depth - 4) % 6 == 0)
        assert ((self._depth - 4) / 6 == 2)

        # вхідний шар мережі
        inputs = Input(shape=self._input_shape)
        # встановлення числа каналів для кожного блока
        # домножуючи їх на параметр широкої залишкової мережі
        # для збільшення каналів
        n_stages = [16, 16 * self._k, 32 * self._k, 64 * self._k]

        # перший згортковий шар мережі
        conv1 = Conv2D(filters=n_stages[0], kernel_size=(3, 3),
                       strides=(1, 1),
                       padding="same",
                       kernel_initializer=self._weight_init,
                       kernel_regularizer=l2(self._weight_decay),
                       use_bias=self._use_bias)(inputs)  # "One conv at the beginning
```

```
(spatial size: 32x32)"

        # створюється залишкові блоки
        block_fn = self._wide_block
        conv2 = self._layer(block_fn, n_input_plane=n_stages[0],
n_output_plane=n_stages[1], stride=(1, 1))(conv1)
        conv3 = self._layer(block_fn, n_input_plane=n_stages[1],
n_output_plane=n_stages[2], stride=(2, 2))(conv2)
        conv4 = self._layer(block_fn, n_input_plane=n_stages[2],
n_output_plane=n_stages[3], stride=(2, 2))(conv3)
        batch_norm = BatchNormalization(axis=self._channel_axis)(conv4)
        relu = Activation("relu")(batch_norm)

        # вихідні шари для класифікації
        pool = AveragePooling2D(pool_size=(8, 8), strides=(1, 1), padding="same")
(relu)
        flatten = Flatten()(pool)
        # вихід для значення статі
        predictions_g = Dense(units=2, kernel_initializer=self._weight_init,
use_bias=self._use_bias,
                              kernel_regularizer=l2(self._weight_decay),
activation="softmax")(flatten)
        # вихід для віку
        predictions_a = Dense(units=101, kernel_initializer=self._weight_init,
use_bias=self._use_bias,  kernel_regularizer=l2(self._weight_decay),
activation="softmax")(flatten)

        model = Model(inputs=inputs, outputs=[predictions_g, predictions_a])
        return model
```

ДОДАТОК З

Аналіз відеопотоку: ідентифікація людей за статтю та віковою групою

Опис програмного модуля

УКР.НТУУ “КПІ ім. Ігоря Сікорського”_ТЕФ_АПЕПС_ТВ6127_20Б 13-1

Аркушів 8

Київ - 2020

АНОТАЦІЯ

У додатку описаний програмний модуль системи, який відповідає за створення архітектури моделі машинного навчання системи. Модель виконує задачу ідентифікації обличчя людини за віком і статтю, отримаючи зображення обличчя з кадрів відеопотоку. Модель — це нейронна мережа класу широка залишкова (згорткова) мережа, архітектура і навчання якої реалізовано з використанням бібліотеки глибокого навчання для Python — Keras.

ЗМІСТ

ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ

Програмне забезпечення розв’язує задачу ідентифікації обличчя за віком і статтю. Програмний модуль містить клас моделі системи, у якому реалізована робота моделі. На вхід до моделі потрапляють дані у вигляді векторизованого представлення зображення — кадра з відео. Для коректної роботи модуля необхідна бібліотека глибокого навчання Keras, що є обгорткою до фреймворка глибокого навчання tensorflow.

ОПИС ЛОГІЧНОЇ СТРУКТУРИ

Програмний модуль реалізує клас нейронної мережі WideResidualNetwork. У класі описані 4 метода, два з них є тандер-методами: __init__ та __call__. WideResidualNetwork.__init__ відповідає за ініціалізацію атрибутів класа при створенні нового екземпляра, а метод WideResidualNetwork.__call__ викликається при виклику класа як метода. При виклику класа покроково реалізується мережа шарів, що починається з вхідного шара Input. Наступні шари об’єднані у блоки. До кожного блоку належить певна кількість згорткових шарів Conv2D, шарів нормалізації BatchNormalization та активації Activation("relu"), які також реалізують тонке з’єднання, що є головною ознакою мереж класа Residual (Залишкового). Останнім блок — це блок класифікації, що складається з шара AveragePooling2D, Flatten та двох повнозв’язних шарів Dense, кожен з яких повертає мітку свого класа — віку або статі.

Метод класа WideResidualNetwork._layer створює два блоки мережі, що відрізняються розмірністю вхідних даних та наявністю згорткового шара з ядром (1, 1) для реалізації швидкого з’єднання.

Метод класа WideResidualNetwork._wide_block реалізує широкий залишковий блок мережі. Блок містить два ланцюжка з шарів Conv2D, BatchNormalization та Activation("relu"), та швидке з’єднання за допомогою функції add.

ВИКОРИСТАНІ ТЕХНІЧНІ ЗАСОБИ

Для створення програмного модуля використовувалася бібліотека глибокого навчання Keras. З модуля бібліотеки models було імпортовано клас Model для створення екземпляра моделі. Для реалізації шарів мережі імпортовано шари з модуля layers: Input, Activation, add, Dense, Flatten, Dropout, Conv2D, AveragePooling2D з підмодуля convolutional, BatchNormalization з підмодуля normalization.

ВИКЛИК І ЗАВАНТАЖЕННЯ

Програмний модуль імпортується у модуль, де буде створено екземпляр моделі, який використовуватиметься для ідентифікації обличчя за віком і статтю. Приклад створення екземпляру: model = WideResidualNetwork(face_size, depth=depth, k=width)().

ВХІДНІ ТА ВИХІДНІ ДАНІ

Вхідними даними для створення екзмепляра класу WideResidualNetwork є параметри моделф — розмірність зображень, глибина мережі та параметр ширини широкої залишкової мережі. При виклику моделі для ідентифікації необхідне векторизоване зображення обличчя людини.

При ідентифікації модель повертає на вихід два значення: віку і статі.